

Napredni razvoj programске potpore za web

**- predavanja -
2022./2023.**

9. Jednostranične web-aplikacije Vue.js

Creative Commons



- slobodno smijete:

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **prerađivati** djelo



- pod sljedećim uvjetima:

- **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.



U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licence preuzet je s <http://creativecommons.org/>

Omogućimo vizualno isticanje odabranog diska, prebacivanje diskova (uz pravila), te „dnevnik“



21. 05. 2021. 11:11:56 No disk may be placed on top of a disk that is smaller than it.
21. 05. 2021. 11:04:48 Selected disk: 2 at rod A
21. 05. 2021. 11:04:47 Moved disk 1 from A to B
21. 05. 2021. 11:04:45 Selected disk: 1 at rod A



Congrats, game over in 7 moves!

That is optimal.

21. 05. 2021. 11:14:00 Moved disk 1 from A to B
21. 05. 2021. 11:13:59 Selected disk: 1 at rod A
21. 05. 2021. 11:13:58 Moved disk 2 from C to B
21. 05. 2021. 11:13:57 Selected disk: 2 at rod C
21. 05. 2021. 11:13:56 Moved disk 1 from C to A
21. 05. 2021. 11:13:55 Selected disk: 1 at rod C
21. 05. 2021. 11:13:54 Moved disk 3 from A to B
21. 05. 2021. 11:13:53 Selected disk: 3 at rod A
21. 05. 2021. 11:13:52 Moved disk 1 from B to C
21. 05. 2021. 11:13:51 Selected disk: 1 at rod B
21. 05. 2021. 11:13:50 Moved disk 2 from A to C

Reactivity

index.html

```
...
<div v-if="isGameOver" class="board card">
  <h1>Congrats, game over in {{ moves }} moves!</h1>
  <h2>That is {{ moves !== (Math.pow(2, diskNumber)-1) ? "SUB" : "" }}optimal.</h2>
</div>
<div v-else class="board card">
  <div class="rod" @click="onSelectTo('A')">
    <div v-for="disk in positionA"
      class="disk"
      :class="{selectedDisk : disk === selectedDisk, inactiveDisk: selectedDisk !== null && disk !== selectedDisk}"
      @click="onSelectFrom(disk, 'A')"
      :style="rodStyle(disk)">
    </div>
  </div>
</div>
```

Analogno za druga dva...

computed property – kada imamo iole složeniju logiku u templateima (koja se temelji na modelu) možemo ju premjestiti u computed property. Npr. izračun optimalnog broja poteza gore ili klase se također mogao „izmjestiti”

v-if je igra gotova, ispišemo je li optimalan broj poteza,
v-else prikazujemo igru (board)

(uvjetno) postavljamo klase selectedDisk i inactiveDisk

hanoi.js

```
var app = new Vue({
  el: '#app',
  data: { diskNumber: 3,
    positionA: [1, 2, 3], positionB: [], positionC: [],
    selectedDisk: null, selectedRod: null,
    logMessages: [],
    moves: 0
  },
  computed: {
    isGameOver: function() {
      return this.positionB.length === this.diskNumber
        || this.positionC.length === this.diskNumber
    }
  }, ... nastavak na sljedećem slajdu...
```

Reactivity: odabir i spuštanje diska

hanoi.js

```

methods: { ...
  onSelectFrom(disk, rod) {
    if (
      (this.positionA[0] === disk)
      || (this.positionB[0] === disk)
      || (this.positionC[0] === disk)
    ) {
      this.selectedDisk = disk;
      this.selectedRod = rod;
      this.log("Selected disk: "
+ this.selectedDisk + " at rod "
+ this.selectedRod);
    } else {
      this.log("You can only select the
topmost disk.");
    }
  },
  log(msg) {
    this.logMessages.unshift({
      ts: new Date().toLocaleString("hr-HR"),
      text: msg,
    });
  },
  ...

```

hanoi.js

```

onSelectTo(rod) {
  if (this.selectedDisk
    && rod !== this.selectedRod) {
    let positionABC = [this.positionA, this.positionB,
                      this.positionC];
    let idxTo = rod.charCodeAt(0) - "A".charCodeAt(0);
    let rodToArray = positionABC[idxTo];
    let idxFrom = this.selectedRod.charCodeAt(0) - "A".charCodeAt(0);
    let rodFromArray = positionABC[idxFrom];

    if (rodToArray.length && rodToArray[0] < this.selectedDisk) {
      this.log(
        "No disk may be placed on top of a disk that is smaller than it."
      );
    } else {
      rodFromArray.shift();
      rodToArray.unshift(this.selectedDisk);
      this.log("Moved disk " + this.selectedDisk + " from " +
        this.selectedRod + " to " + rod
      );
      this.selectedRod = null;
      this.selectedDisk = null;
      this.moves++;
    }
  }
}

```

Ispis dnevnika i CSS

index.html

```
...
<div class="log card">
  <div v-for="msg in logMessages" >
    <span class="log--time">
      {{ msg.ts }}
    </span>
    <span class="log--message">
      {{ msg.text }}
    </span>
  </div>
</div>
```

hanoi.css

```
.selectedDisk {
  margin: 5px;
  box-shadow: 0px 0px 5px 5px rgba(0, 255, 0, 0.5),
    0px 0px 5px 5px inset rgba(255, 238, 0, 0.5);
}
.inactiveDisk {
  box-shadow: 0px 0px 20px 10px inset rgba(0, 0, 0, 0.7);
}
div.log {
  margin-top: 20px;
  max-height: 200px;
  overflow: auto;
  min-height: 100px;
  font-family: Verdana, Geneva, Tahoma, sans-serif;
}
.log--time {
  font-size: 0.9em;
  background-color: honeydew;
  padding: 0 5px 0 5px;
  border: 1px solid gray;
  border-radius: 2px;
}
```

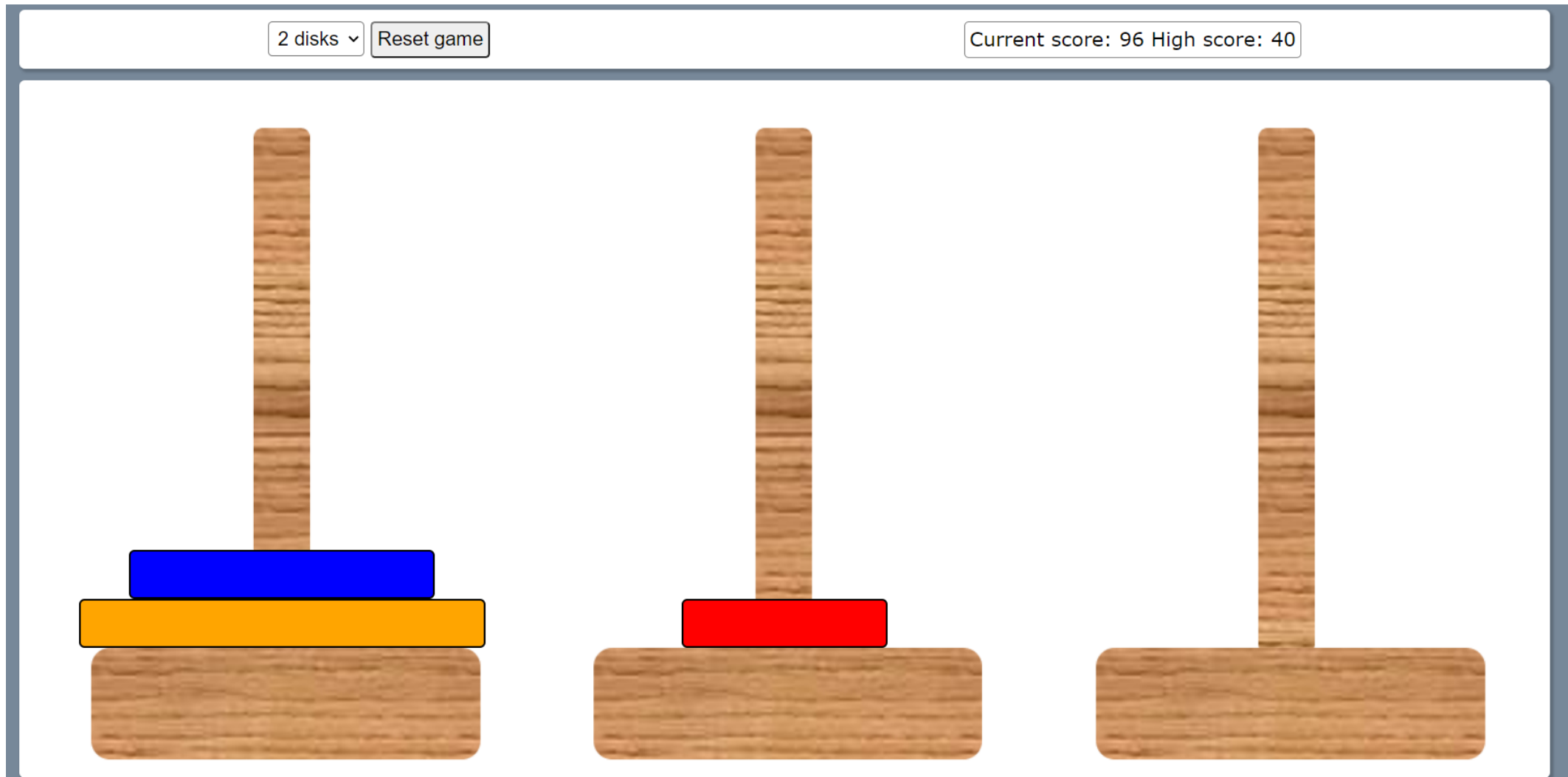
21. 05. 2021. 11:14:00	Moved disk 1 from A to B
21. 05. 2021. 11:13:59	Selected disk: 1 at rod A
21. 05. 2021. 11:13:58	Moved disk 2 from C to B
21. 05. 2021. 11:13:57	Selected disk: 2 at rod C
21. 05. 2021. 11:13:56	Moved disk 1 from C to A
21. 05. 2021. 11:13:55	Selected disk: 1 at rod C
21. 05. 2021. 11:13:54	Moved disk 3 from A to B
21. 05. 2021. 11:13:53	Selected disk: 3 at rod A
21. 05. 2021. 11:13:52	Moved disk 1 from B to C
21. 05. 2021. 11:13:51	Selected disk: 1 at rod B
21. 05. 2021. 11:13:50	Moved disk 2 from A to C

■ Par napomena uz *computed property*

- Sličnost s methods
 - Kada koristiti jedno, kada drugo?
 - **Computed properties** - kada samo mijenjamo prezentaciju, ali ne i podatke
 - Štoviše - **paziti da ne mijenjamo podatke unutra *computed properties*** - nezgodni bugovi!
pogledajte: <https://vueschool.io/lessons/computed-properties-in-vue-3?friend=vuejs>
 - **Methods** - kada mijenjamo podatke
 - Bitna razlika - CP su **keširani!**
 - Ponovo se izračunavaju samo ako se promijeni neka **reaktivna** varijabla na temelju koje se izračunava
 - Npr. nikad se neće osvježiti:
 - (moguće isključiti cache)
- Postoji i [Computed setter](#)

```
computed: {  
  now() {  
    return Date.now()  
  }  
}
```

Omogućimo novu igru s različitim brojem diskova, izračunajmo score i zapamtimo ga kako se ne bi izgubio nakon osvježavanja stranice



Dodajemo highScore, učítavamo, snimamo...

index.html

hanoi.js

```

<div class="board card">
  <div class="reset-game">
    <select ref="disksNumber"
      class="form-control">
      Number of disks:
      <option value="2">2 disks</option>
      <option value="3">3 disks</option>
      <option value="4">4 disks</option>
      <option value="5">5 disks</option>
    </select>
    <button @click="resetGame"
      class="form-control">Reset game
    </button>
  </div>
  <div class="high-score"
    :class="{ newHighScore: isNewHighScore }">
    Current score: {{score}}
    High score: {{highScore}}
  </div>
</div>

```

Lifecycle event

```

computed: { ...
  score: function () {
    return this.diskNumber * 10 -
      11 * (this.moves - this.optimalMoves());
  },
  isNewHighScore: function () {
    return this.isGameOver
      && (this.score > this.highScore);
  }
}, methods: { ...
  onSelectTo(rod) {
    ...
    this.moves++;
    if (this.isNewHighScore) {
      this.highScore = this.score;
      localStorage.setItem('highScore', this.highScore);
      this.log("*** CONGRATS!!! NEW HIGH SCORE!! **");
    }
  },
  beforeMount() {
    let highScore = localStorage.getItem('highScore');
    if (highScore !== null) {
      this.highScore = parseInt(highScore);
    } ...

```



Vue.js

a crash course...

routing, components, forms, Pinia

Inicijalizacija Vue aplikacije

<https://vuejs.org/guide/quick-start.html#creating-a-vue-application>

```
λ npm init vue@latest
Need to install the following packages:
  create-vue@latest
Ok to proceed? (y) y

Vue.js - The Progressive JavaScript Framework

✓ Project name: ... fer-demo-spa
✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? ... No / Yes
✓ Add Prettier for code formatting? ... No / Yes

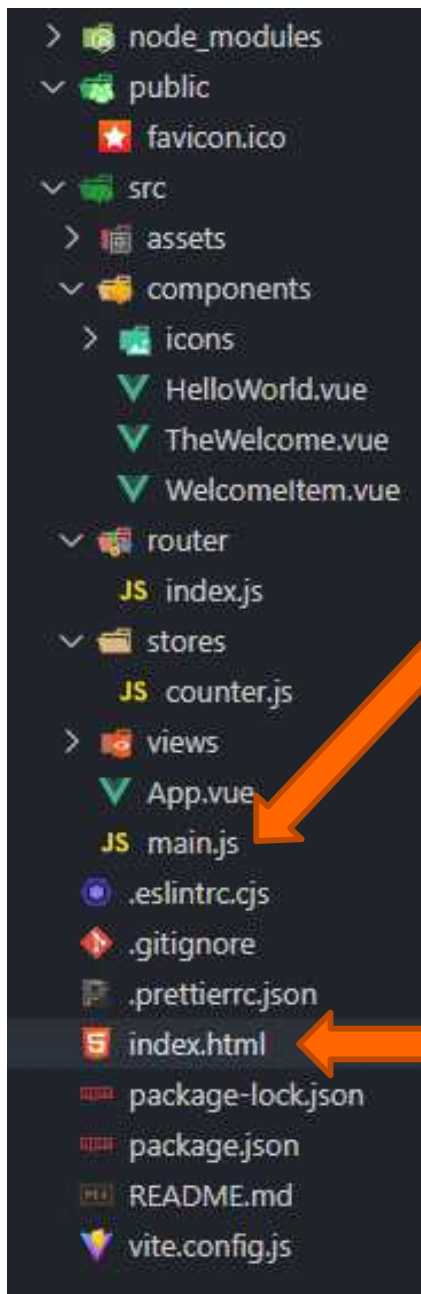
Scaffolding project in C:\web2\fer-demo-spa...

Done. Now run:

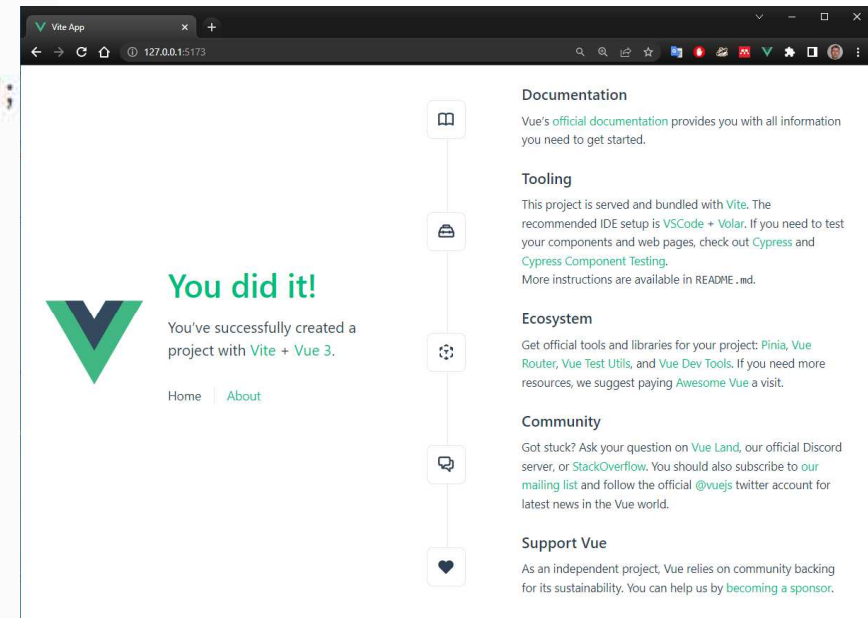
  cd fer-demo-spa
  npm install
  npm run lint
  npm run dev
```

ive - Tehnička nastava Web1 - Razvoj programske potpo

Struktura projekta



```
src > JS main.js > ...
1 import { createApp } from "vue";
2 import { createPinia } from "pinia";
3
4 import App from "./App.vue";
5 import router from "./router";
6
7 import "./assets/main.css";
8
9 const app = createApp(App);
10
11 app.use(createPinia());
12 app.use(router);
13
14 app.mount("#app");
```



```
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <link rel="icon" href="/favicon.ico">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Vite App</title>
8   </head>
9   <body>
10    <div id="app"></div>
11    <script type="module" src="/src/main.js"></script>
12  </body>
13 </html>
```

Single-File Components (.vue datoteke)

```
<template>
  (ovdje pišemo HTML)
</template>

<script>
  (ovdje pišemo JS)
</script>

<style scoped>
  (ovdje pišemo CSS)
</style>
```

<https://vuejs.org/guide/scaling-up/sfc.html>

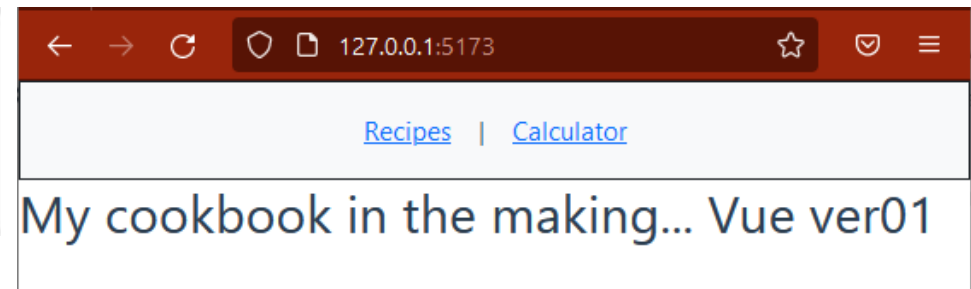
- Imamo tri sekcije:
 - `template`
 - `script`
 - `style`
- Vite automatski prevodi u JS
 - <https://vitejs.dev/guide/why.html>
- Koristi se:
 - View
 - Component

View vs Component

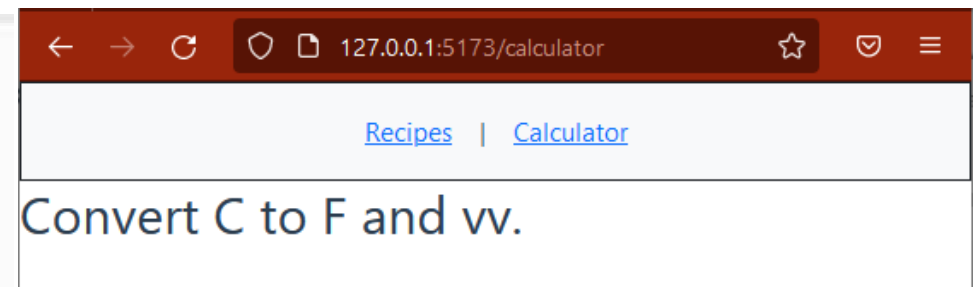
- Primjećujemo dva koncepta u ustroju projekta (i dva različita direktorija):
 - **Views** (src/views)
 - **Components** (src/components)
- Na tehničkoj razini su to iste stvari – **komponente**
 - Vrijedi sve s prethodnog slajda
- Ali semantički:
 - **Views** predstavljaju **stranice** naše **više-stranične** aplikacije i tipično se referenciraju iz usmjerivača (router, uskoro)
 - **Components** predstavljaju (ponovo upotrebljive) komponente koje se koriste na N stranica
- Postoje **različite konvencije** kako ih nazivati i gdje ih pohranjivati (npr. *pages*, *views*, itd.)

Izmijenimo generirani projekt u začetak naše Cookbook aplikacije

```
src > views > RecipesView.vue > ...  
1 <template>  
2 | <h1>My cookbook in the making... Vue ver01</h1>  
3 </template>
```



```
src > views > CalculatorView.vue > ...  
1 <template>  
2 | <h1>Convert C to F and vv.</h1>  
3 </template>
```



Ako koristite **VS Code**, instalirati **Volar**
~~Vetur~~ za *syntax highlighting* i sl.

Konfiguracija routera

main.js

```
import router from "./router";  
  
...  
app.use(router);  
  
...
```

App.vue

```
<script setup>  
import { RouterLink, RouterView } from "vue-router";  
</script>  
<template>  
  <header>  
    <nav class="d-flex justify-content-center bg-light border border-bottom border-dark p-3 w-100">  
      <RouterLink to="/">Recipes</RouterLink>  
      |  
      <RouterLink  
to="/calculator">Calculator</RouterLink>  
    </nav>  
  </header>  
  <RouterView />  
</template>  
<style scoped>  
nav a {  
  margin-left: 1rem; margin-right: 1rem;  
}  
</style>
```

router/index.js

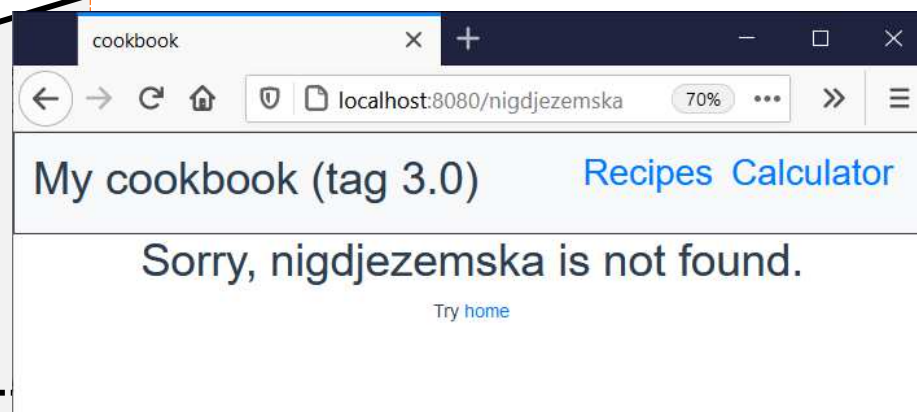
```
import { createRouter, createWebHistory } from "vue-router";  
import RecipesView from "../views/RecipesView.vue";  
import CalculatorView from  
  "../views/calculatorView.vue";  
const router = createRouter({  
  history: createWebHistory(import.meta.env.BASE_URL),  
  routes: [  
    {  
      path: "/",  
      name: "recipes",  
      component: RecipesView,  
    },  
    {  
      path: "/calculator",  
      name: "calculator",  
      component: CalculatorView,  
    },  
  ],  
});  
export default router;
```


Dynamic routing, catch-all

router/index.js

```
...  
const routes = [ // postoji i alias: "/"  
  { path: "/", // mogli smo i redirect: "/recipes"  
    component: Recipes,  
  },  
  {  
    path: "/recipes/:id?",  
    component: Recipes,  
  },  
  { path: "/calculator",  
    component: Calculator,  
  },  
  { path: "/*",  
    name: "NotFound",  
    component: NotFound,  
  },  
];  
...
```

Opcionalni (zbog upitnika) parametar, poslije dostupan kao `$route.params.id`



NotFound.vue

```
<template>  
  <h1>Sorry, {{ $route.params.catchAll }}  
    is not found.</h1>  
  Try <router-link to="/">home</router-link>  
</template>
```

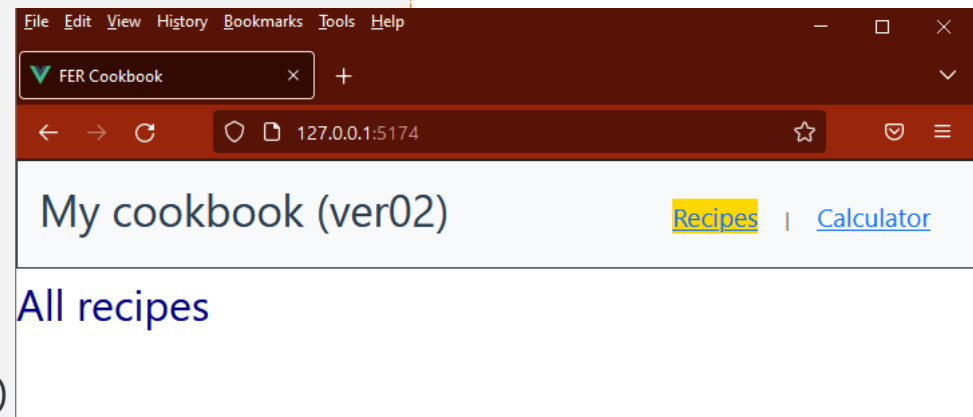
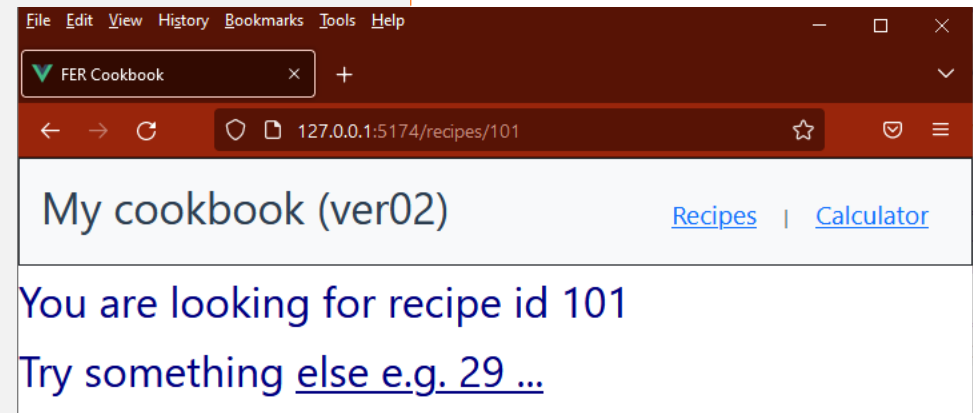
catchAll je proizvoljno ime varijable, u zagradi je regex

Recipes.vue 1/2

Recipes.vue

```
<template>
  <div v-if="$route.params.id">
    You are looking for recipe id {{ $route.params.id }}
    <br>Try something <router-link :to="'/recipes/' + somethingElse">
      else e.g. {{ somethingElse }} ...
    </router-link>
  </div>
  <div v-else>All recipes</div>
  <ul>
    <li v-for="msg in routeChanges"
      :key="msg">{{ msg }}</li>
  </ul>
</template>
<script>
export default {
  data() {
    return {
      routeChanges: [],
      somethingElse: Math.floor(Math.random() * 1000)
    };
  }, ...
}
```

Primijetiti dvotočku – izraz pod navodnicima se onda tumači kao js expression



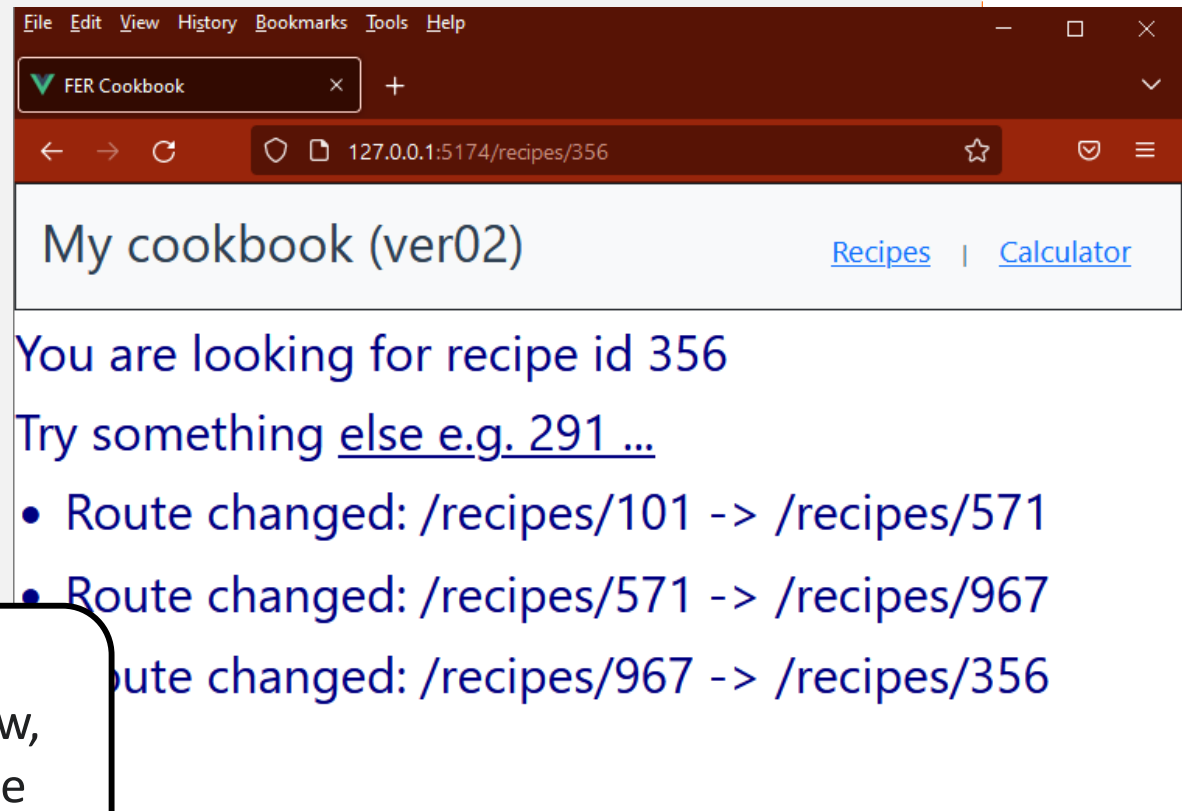
Recipes.vue 2/2: watchers, scoped style

Recipes.vue

```
...  
watch: {  
  $route(to, from) {  
    this.routeChanges.push(`Route changed: ${from.path} -> ${to.path} `);  
    this.somethingElse = Math.floor(Math.random() * 1000);  
  },  
},  
};  
</script>  
  
<style scoped>  
* {  
  font-size: 32px;  
  color: navy;  
}  
</style>
```

watch opcija – moguće je pratiti promjene neke varijable!
Inače ne bi detektirali promjenu rute, jer Vue čuva komponentu u memoriji

Primijetiti da je stil primijenjen samo na view, npr. naslov je i dalje crne boje, zašto?



Što će se dogoditi ako ručno upišemo npr. /recipes/101 ?

- Uočiti razliku:

Ako ga ne konfiguriramo, WS će vjerojatno vratiti 404.

Npr. probajte npm run build i onda dist folder pogledati preko LiveServera



GET /recipes/101

Web server

NotFound.vue

```
<template>
...
Try <router-link to="/recipes/101">
  recipe 101
</router-link>
</template>
```

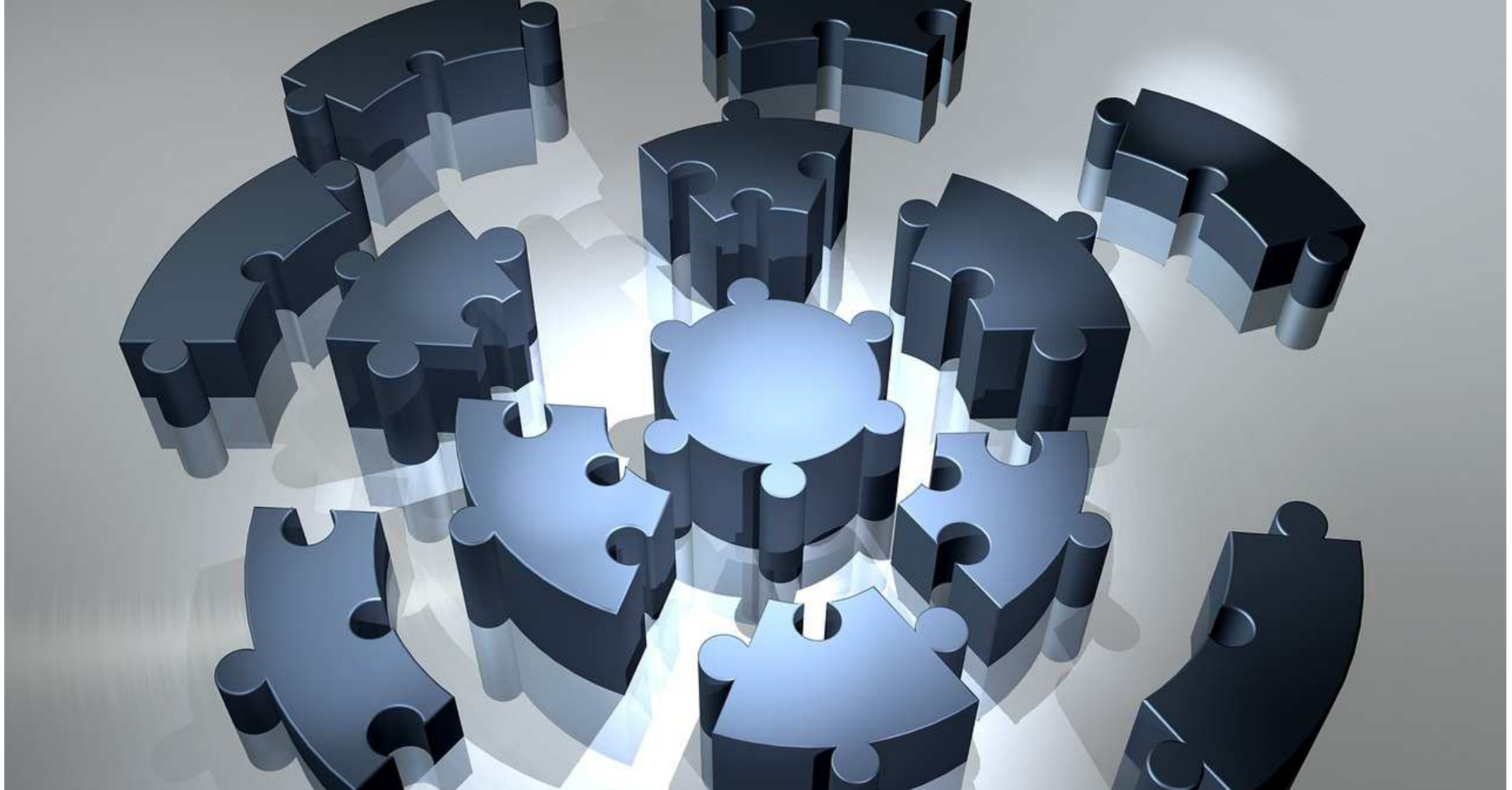
DM1

ovo je komentar nastavno na zadatak za vježbu? zato sto je slajd izmedju zadatka za vjezbu i komponente

Danijel Mlinarić; 15.9.2021.

Zadatak za vježbu

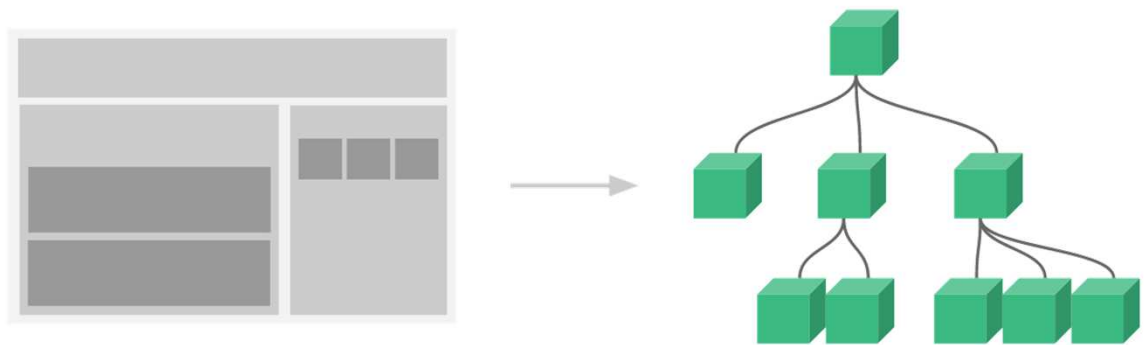
Pogledajte *programmatic navigation*, te dodajte na `Recipes.vue` *button* koji će korisnika prebaciti na **slučajni recept**



Komponente

Komponente

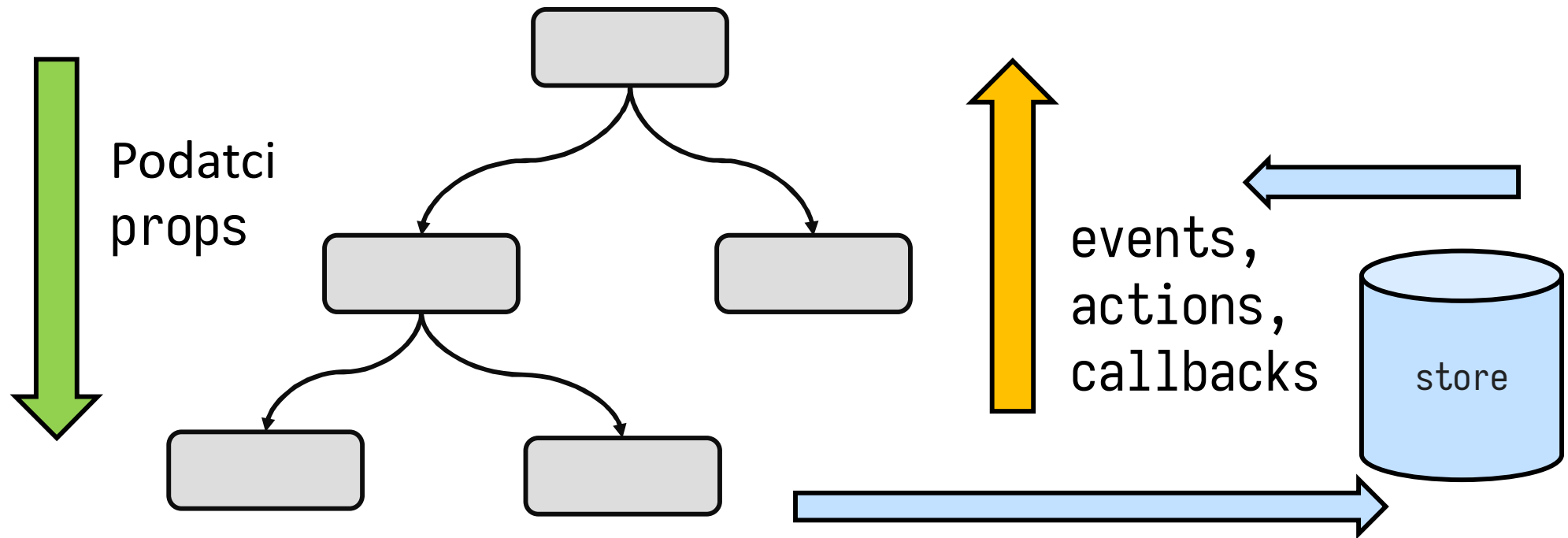
- Kada:
 - Imamo dijelove stranice/markupa koji se ponavljaju
 - Želimo razložiti problem/aplikaciju u manje probleme/aplikacije
- Komponenta je poput mini-aplikacije koja se onda pridijeli aplikaciji
 - Pridjeljujemo im oznaku (*tag*), tipično dvije riječi odvojene crticom kako bi izbjegli preklapanje s HTML oznakama
- Uobičajeno se aplikaciju ustroji u stablo ugniježđenih komponenti,
npr. zaglavlje, tijelo, podnožje, popup/dialog, ...



<https://v3.vuejs.org/guide/component-basics.html>

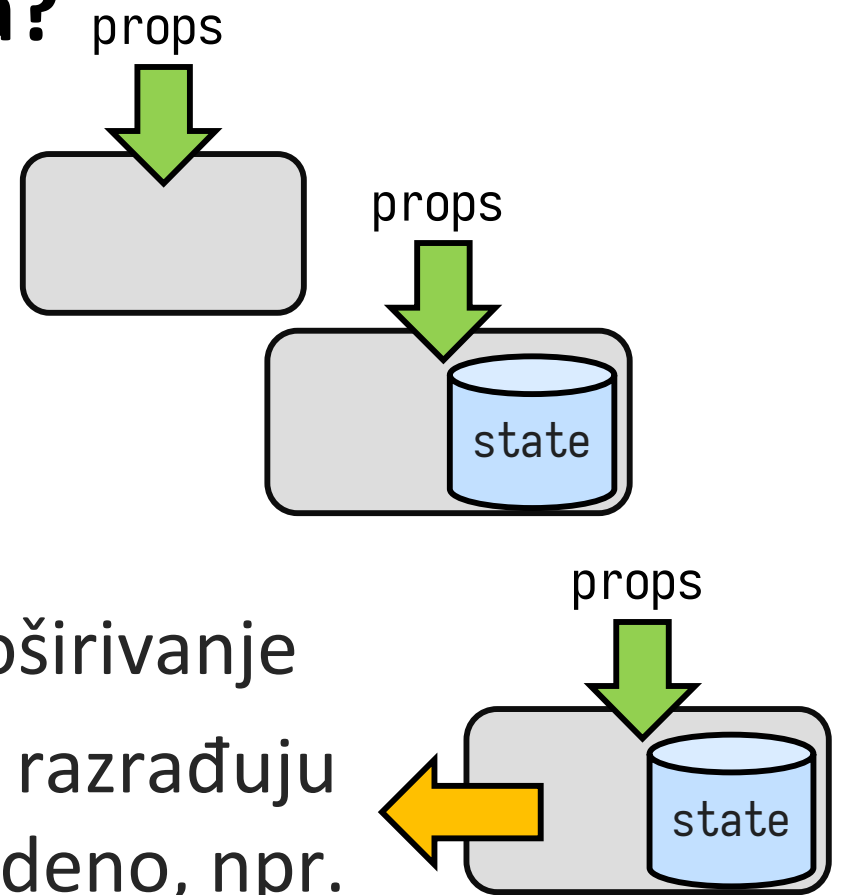
Obrasci komunikacije

- **Roditelj -> dijete**
 - Tipično se komponentama predaju vrijednosti
- **Dijete -> roditelj**
 - Rjeđe, ali ponekad nužno
 - Npr. React ima „*one-way data flow parent → child*”, ali...



Kakva može biti komponenta?

- Tri obilježja:
 - Što prima?
 - Ima li vlastito stanje?
 - Surađuje li s drugima?
 - Po pitanju podataka
 - Omoguće li gniježđenje, proširivanje
- Različiti radni okviri više ili manje razrađuju nomenklaturu s obzirom na navedeno, npr.
 - React ima *class* i *functional* components
 - Angular: *smart* i *dumb/presentational/pure* komponente (samo primaju podatke i iscrtavaju)



Pretvorimo karticu recepta i kalkulator u komponente

ver03

- Prvo jednostavniju – Calculator
 - Vlastito stanje, ne prima parametre, ne surađuje

main.js

```
import { createApp } from "vue";
import App from "./App.vue";
import router from "./router";
import RecipeCard from './components/RecipeCard.vue';
import TempConverter from './components/TempConverter.vue';
```

```
const app = createApp(App);
app.use(router);
app.component('recipe-card', RecipeCard);
app.component('temp-converter', TempConverter);
app.mount("#app");
```

Konvencija: xxx-yyy

Prijavljujemo ih „globalno” čime postaju dostupne u cijeloj aplikaciji.
(sljedeći slajd)

Views/CalculatorView.vue

```
<template>
  <div>
    <h2>Calculator</h2>
    <div class="container-fluid p-2 d-
flex justify-content-center">
      <temp-converter></temp-converter>
    </div>
  </div>
</template>
```

Vue komponente - komentar

- Da bi koristili komponente, moramo ih registrirati
- Dvije mogućnosti:
 1. Globalno registrirati komponentu -> vidljiva svugdje
 - Nedostatci:
 - Onemogućen *treeshaking*
 - Teži uvid u ovisnost/povezanost komponenti
 2. Lokalno registrirati komponentu
 - Vidljivo samo u toj komponenti, ne i njenoj djeci

```
import { createApp } from 'vue'  
  
const app = createApp({})  
  
app.component(  
  // the registered name  
  'MyComponent',  
  // the implementation  
  {  
    /* ... */  
  }  
)
```

If using SFCs, you will be registering the imported `.vue` files:

```
import MyComponent from './App.vue'  
  
app.component('MyComponent', MyComponent)
```

TempConverter.vue

ver03

- Prvo jednostavniju – Calculator
 - Ima vlastito stanje, ne prima parametre, ne surađuje

Svaki dio se može izostaviti, npr. ovdje nema style

struktura:

```
<template>
...
</template>
<script >
...
</script >
<style>
...
</style>
```

components/TempConverter.vue

```
<template>
  <div><h3>Temperature converter</h3>
    <form>
      <div><label>Celsius</label>
      <div><input
        v-model.number="tempCelsius"
        @keyup="c2f"/>
      </div>
    </div>
    <div><label>Fahrenheit</label>
    <div><input
      v-model.number="tempFahrenheit"
      @keyup="f2c"
    />
    </div></div>
  </form></div>
</template>
```

Izbačeni svi stilovi i sitnice koje nisu u fokusu, pogledati src

components/TempConverter

```
<script>
export default {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32,
    };
  },
  methods: {
    c2f() {
      this.tempFahrenheit = Math.round((this.tempCelsius * 9) / 5 + 32);
    },
    f2c() {
      this.tempCelsius = Math.round(((this.tempFahrenheit - 32) * 5) / 9);
    },
  },
};
</script>
```

■ Domaća zadaća

- Primijenite komponente, dodajte joj „Log it” gumb koji smo već vidjeli
- Listu temperatura možete prikazati kao tooltip ili sl.
- Time će naša komponente proširiti svoje lokalno stanje (pored temperatura C i F)
- Stavite nekoliko istih komponenti na npr. tu istu stranicu
- Što se događa kada otidete na stranicu recepata i vratite se?
 - Pogledajte i keep-alive
 - Može se keširati i komponenta i stranica, pazite

<https://stackoverflow.com/questions/40898440/vue-2-keep-alive-not-working-with-router-view-and-key>

Vue props

- Pogledajmo: <https://v3.vuejs.org/guide/component-props.html>

- **Static:**

```
<blog-post title="My journey with Vue"></blog-post>
```

- **Dynamic:**

```
<!-- Dynamically assign the value of a variable -->
<blog-post :title="post.title"></blog-post>

<!-- Dynamically assign the value of a complex expression -->
<blog-post :title="post.title + ' by ' + post.author.name"></blog-post>
```

- Mogu se predavati različiti tipovi: number, boolean, array,...

- **Object:**

- Itd. pogledati dokumentaciju

```
<!-- Even though the object is static, we need v-bind to tell Vue that -->
<!-- this is a JavaScript expression rather than a string. -->
<blog-post
  :author="{
    name: 'Veronica',
    company: 'Veridian Dynamics'
  }"
></blog-post>

<!-- Dynamically assign to the value of a variable. -->
<blog-post :author="post.author"></blog-post>
```

Citat iz dokumentacije:

One-Way Data Flow

All props form a **one-way-down binding** between the child property and the parent one: when the parent property updates, it will flow down to the child, but not the other way around. This prevents child components from accidentally mutating the parent's state, which can make your app's data flow harder to understand.

In addition, every time the parent component is updated, all props in the child component will be refreshed with the latest value. This means you should **not** attempt to mutate a prop inside a child component. If you do, Vue will warn you in the console.

There are usually two cases where it's tempting to mutate a prop:

1. The prop is used to pass in an initial value; the child component wants to use it as a local data property afterwards. In this case, it's best to define a local data property that uses the prop as its initial value:

```
1  props: ['initialCounter'],  
2  data() {  
3    return {  
4      counter: this.initialCounter  
5    }  
6  }
```

2. The prop is passed in as a raw value that needs to be transformed. In this case, it's best to define a computed property using the prop's value:

```
1  props: ['size'],  
2  computed: {  
3    normalizedSize() {  
4      return this.size.trim().toLowerCase()  
5    }  
6  }
```

Note

Note that objects and arrays in JavaScript are passed by reference, so if the prop is an array or object, mutating the object or array itself inside the child component **will** affect parent state.

RecipeCard.vue 1/2 (script, style)

- Nema stanje, **prima parametre**, ne surađuje

components/RecipeCard.vue

```
<script>
export default {
  props: [
    "id", "image", "name", "description", "cookTime",
    "prepTime", "recipeYield", "datePublished", "url",
    "ingredients"
  ],
  computed: {
    idUrl() {
      return '/recipes' + this.id;
    }
  }
};
</script>
<style scoped>
  div.card-body .badge {
    white-space: pre-wrap;
  }
</style>
```

Template na sljedećem
slajdu, uočiti props

Scoped – lokalni stil,
utječe samo na ovu
komponentu!

views/RecipesView.vue

```
<template>
  <div v-if="selectedRecipe">
    Recipe {{{ id }}}
    <recipe-card :key="selectedRecipe.id"
      v-bind="selectedRecipe"
      :zoom="1"
    ></recipe-card>
  </div>
  <div v-else>
    <h2>All recipes ({{{allRecipes.length}}})</h2>
    <div>
      <recipe-card v-for="recipe in allRecipes"
        :key="recipe.id"
        v-bind="recipe"
      ></recipe-card>
    </div>
  </div>
</template>
```

Static
property

„If you want to pass all
the properties of an object
as props, you can use v-
bind without an argument
(v-bind instead of :prop-
name).”

RecipeCard.vue 2/2 (template)

components/RecipeCard.vue

```
<template>
  <div class="card mt-2 mr-2" :style="{ width: 230 * (zoom ? 3 : 1) + 'px' }">
    <router-link :to="'/recipes/' + id" class="mr-3">
      
    </router-link>
    <div class="card-body">
      <h5 class="card-title">{{ name }}</h5>
      <p class="card-text">{{ description }}</p>
      <ul class="list-group">
        <li class="list-group-item">
          Cook/prep time: {{ cookTime }}/{{ prepTime }}
        </li>
        <li class="list-group-item">Yield: {{ recipeYield }}</li>
        <li class="list-group-item">Published: {{ datePublished }}</li>
        <li class="list-group-item"><a :href="url" target="_blank" class="card-link">{{ url.substring(0, 20) }}...</a></li>
      </ul>
    </div>
    <details v-if="ingredients">
      <summary><h3>Ingredients</h3></summary>
      <ol class="list-group">
        <li
          v-for="ingredient in ingredients"
          :key="ingredient"
          class="list-group-item"
        >
          {{ ingredient }}
        </li>
      </ol>
    </details>
  </div>
</template>
```

Ako je zoom postavljen, koristimo širu karticu

Nema ničeg posebno novog, jednostavno koristimo props kao normalne varijable kod definiranja obrasca odnosno iscrtavanja

← → ↺

📄 127.0.0.1:5173


70% ☆

🔍 ☰

My cookbook (ver03)

Recipes | Calculator

All recipes (96)



Baked Polenta Fries Recipe


Great party finger food. A thick slab of polenta is sliced into the shape of a french fry and baked off.

Cook/prep time: /

Yield: Makes 2 dozen wide-cut fries.

Published: 2006-08-13

<http://www.101cookbo...>



Heirloom Tomato Tart in a Parmesan Crust


A knock-out tomato tart recipe - it highlights the flavor and vibrancy of the tomatoes while remaining mush-free.

Cook/prep time: /

Yield:

Published: 2005-08-05

<http://www.101cookbo...>



Thinnest Oatmeal Cookies


One of the easiest cookies I know how to make - made from rolled oats, they're razor thin and lacy, golden, flecked with poppy seeds, with an anise accent from crushed fennel seeds.

Cook/prep time: PT10M/PT10M

Yield: Makes about 2 dozen cookies.

Published: 2012-12-14

<http://www.101cookbo...>



Whole Vanilla Bean Cookies


Snappy, small, fragrant, vanilla wafer cookies made with a whole vanilla pod. The entire thing!

Cook/prep time: PT15M/PT45M

Yield: Makes about 2 dozen small cookies.

Published: 2012-07-18

<http://www.101cookbo...>



Baked Quinoa Patties


A few London pics, a look at how I packed my lunch for the flight, and the baked quinoa patties I brought along with me - dill, feta, chives, cumin, and garlic.

Cook/prep time: PT25M/PT10M

Yield: Makes about a dozen patties.

Published: 2011-10-11

<http://www.101cookbo...>



Whole Wheat Chocolate Chip Skillet Cookie


Made using 100% whole wheat flour and hand-chopped chocolate chips, this is a skillet-baked twist on Kim Boyce's celebrated Chocolate Chip Cookies.

Cook/prep time: PT30M/PT10M

Yield:

Published: 2011-02-08

<http://www.101cookbo...>



Maple Huckleberry Coffee Cake Recipe


A wonderful coffee cake recipe made from fresh wild huckleberries using maple syrup as a sweetener instead of white sugar.

Cook/prep time: /

Yield: Serves 12 - 16 modest slices.

Published: 2008-09-22

<http://www.101cookbo...>



Cherry Cobbler Recipe

A rustic, cherry cobbler recipe made from fresh cherries - though you can certainly try this recipe with other types of summer fruit and berries.

Cook/prep time: /

Yield: Serves about 8.

Published: 2008-07-16

<http://www.101cookbo...>

Ingredients

Ingredients

Ingredients

Ingredients

Ingredients

Ingredients

Ingredients

File Edit View History Bookmarks Tools Help

FER Cookbook

← → ↺

📄 127.0.0.1:5173/recipes/85


70% ☆

🔍 ☰

My cookbook (ver03)

Recipes | Calculator

Recipe #85



Great Chocolate Chip Cookie Recipe

If you like a serious chocolate chip cookie, this is the recipe. A high chip to dough ratio guarantees lots of chocolate in every bite, and the walnuts add crunch, density, and a delicious flavor to the overall mix. They are that good.

Cook/prep time: /

Yield:

Published: 2005-04-11

<http://www.101cookbo...>

Ingredients

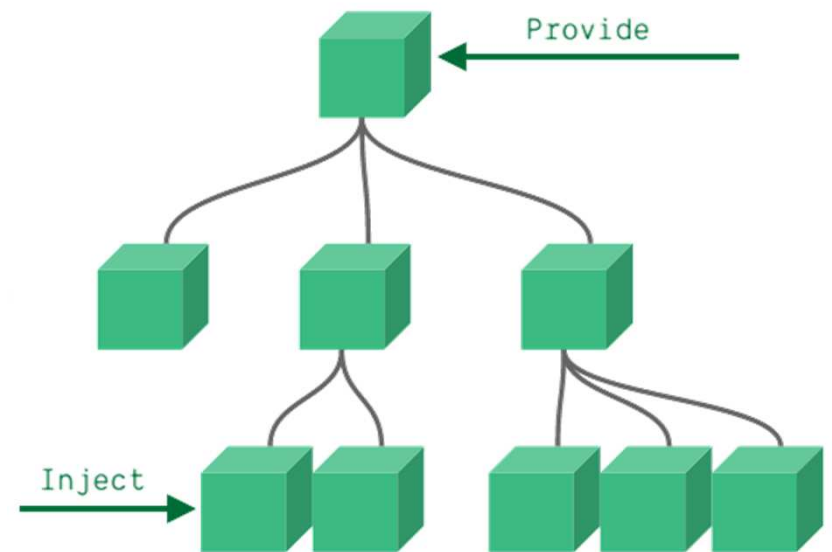
UNIZG-FER

Napredni razvoj programske potpore za web

34

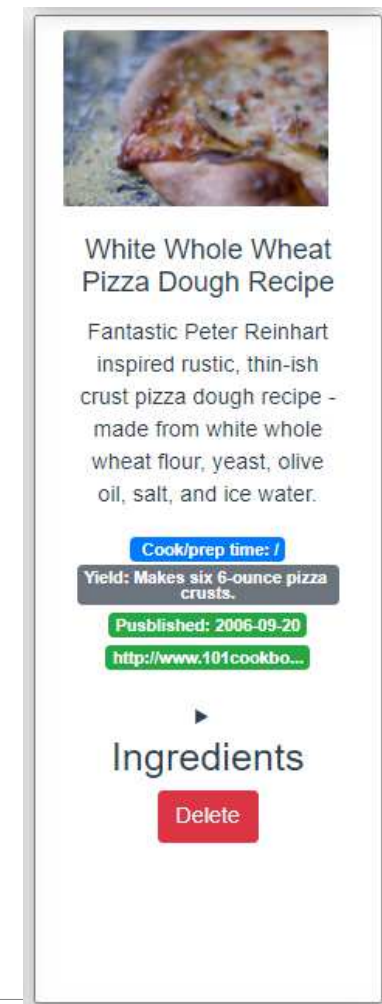
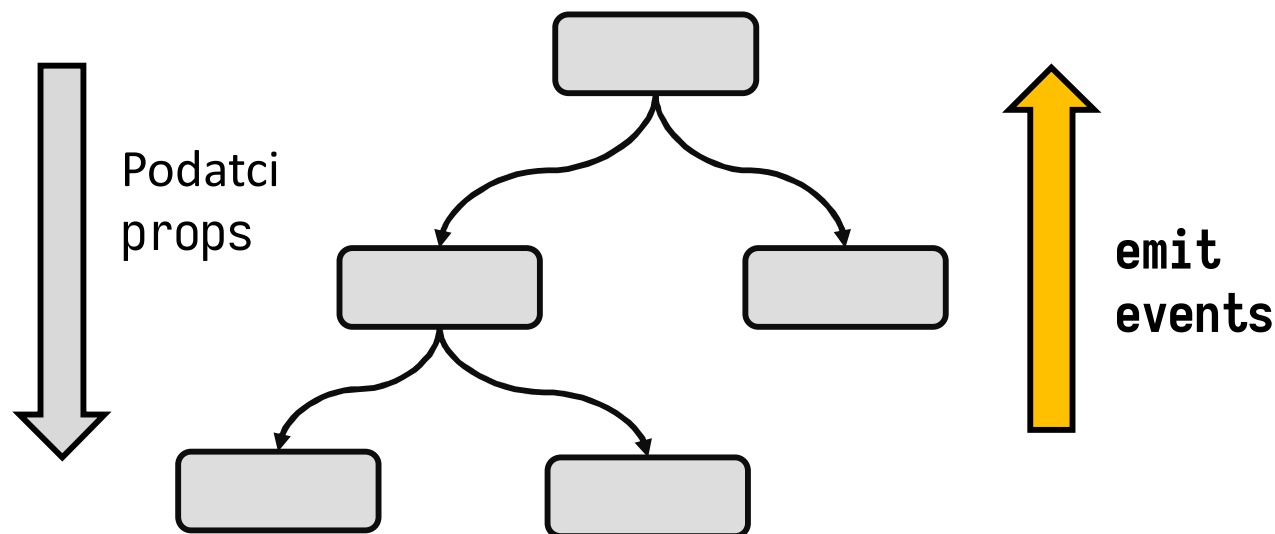
Provide/inject

- <https://v3.vuejs.org/guide/component-provide-inject.html>
- Ako imamo duboku hijerarhiju komponenti i želimo poslati svojstvo (*prop*) ne samo neposrednom djetetu onda to može biti zamorno („štafeta”)
- *provide*: roditelj pruža svojstvo svim svojim nasljednicima (cijelo podstablo)
- *inject*: dijete prijavi što koristi
- „*long-range props*”:
 - Roditelji ne moraju znati tko koristi
 - Djeca ne moraju znati odakle dolazi
- Moguće i [App-level Provide](#)



Emitiranje događaja

- „Suprotni smjer” – ponekad je potrebno iz komponente nešto javiti roditelju
- Dodajmo u recipe-card gumb za brisanje recepta
 - Komponenta **emitira** događaj brisanja svom roditelju
 - Roditelj se povezuje na emitirane događaje i poduzima odgovarajuće radnje – ovdje izbacuje element iz polja



Emitiranje događaja „deleteRecipe”

components/RecipeCard.vue

```
<template>
  <small-card>
    (... isto kao prije ...)
    <button class="btn btn-danger"
      @click="deleteRecipe">
      Delete
    </button>
  </small-card>
</template>
<script>
export default {
  emits: ["deleteRecipe"],
  (... )
  methods: {
    deleteRecipe() {
      this.$emit('deleteRecipe', {id: this.id});
    }
  }
};
</script>
```

Na sljedećem slajdu

1

2

views/RecipesView

```
<template>
  <div v-if="selectedRecipe">
    <h2>Recipe #{{ id }}</h2>
    <recipe-card :key="selectedRecipe"
      v-bind="selectedRecipe"
      @delete-recipe="deleteRecipe"
    ></recipe-card>
  </div></div>
  <div v-else>
    <h2>All recipes ({{allRecipes.length}})</h2><hr><div>
      <recipe-card v-for="recipe in allRecipes"
        :key="recipe.id" v-bind="recipe"
        @delete-recipe="deleteRecipe"
      ></recipe-card>
    </div></div>
</template>
<script>...
methods: {
  deleteRecipe (args) {
    this.allRecipes = this.allRecipes.filter(x => x.id !== args.id);
    if (this.selectedRecipe && this.selectedRecipe.id === args.id) {
      this.selectedRecipe = null;
    }
  }
  ...
}
</script>
```

3

„Like components and props, event names provide an automatic case transformation. If you emit an event from the child component in camel case, you will be able to add a kebab-cased listener in the parent”

Utori (*Slots*)

<https://v3.vuejs.org/guide/component-slots.html>

- Komponente mogu definirati mjesto (utor, slot) u koje će se ugraditi drugi sadržaj (string, HTML, druge komponente...)
- Mi ćemo definirati „karticu” – standardni element našeg korisničkog sučelja
 - u nju možemo stavljati razne sadržaje (recept, kalkulator)

components/SmallCard.vue


```
<template>
  <div>
    <slot></slot>
  </div>
</template>
<style scoped>
  div {
    border: 1px gray solid;
    border-radius: 3px;
    width: 250px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
    padding: 10px;
    margin: 5px;
  }
</style>
```

Temperature converter

Celsius

Fahrenheit

Prijavimo u main.js kao i druge komponente.
Postoje i napredne opcije,
npr. *scoped slots*, *named-slots*, ZOKŽNV



Homemade and All-natural Thin Mint Recipe
As promised, an all-natural Thin Mint recipe for you. No shortening, no trans-fats from partially hydrogenated vegetable oils - just good old-fashioned butter, cocoa, vanilla, sugar, chocolate, whole grain flour, and peppermint turned into delicious, thin minty goodness.

Cook/prep time: /

Yield:

Published: 2006-03-12

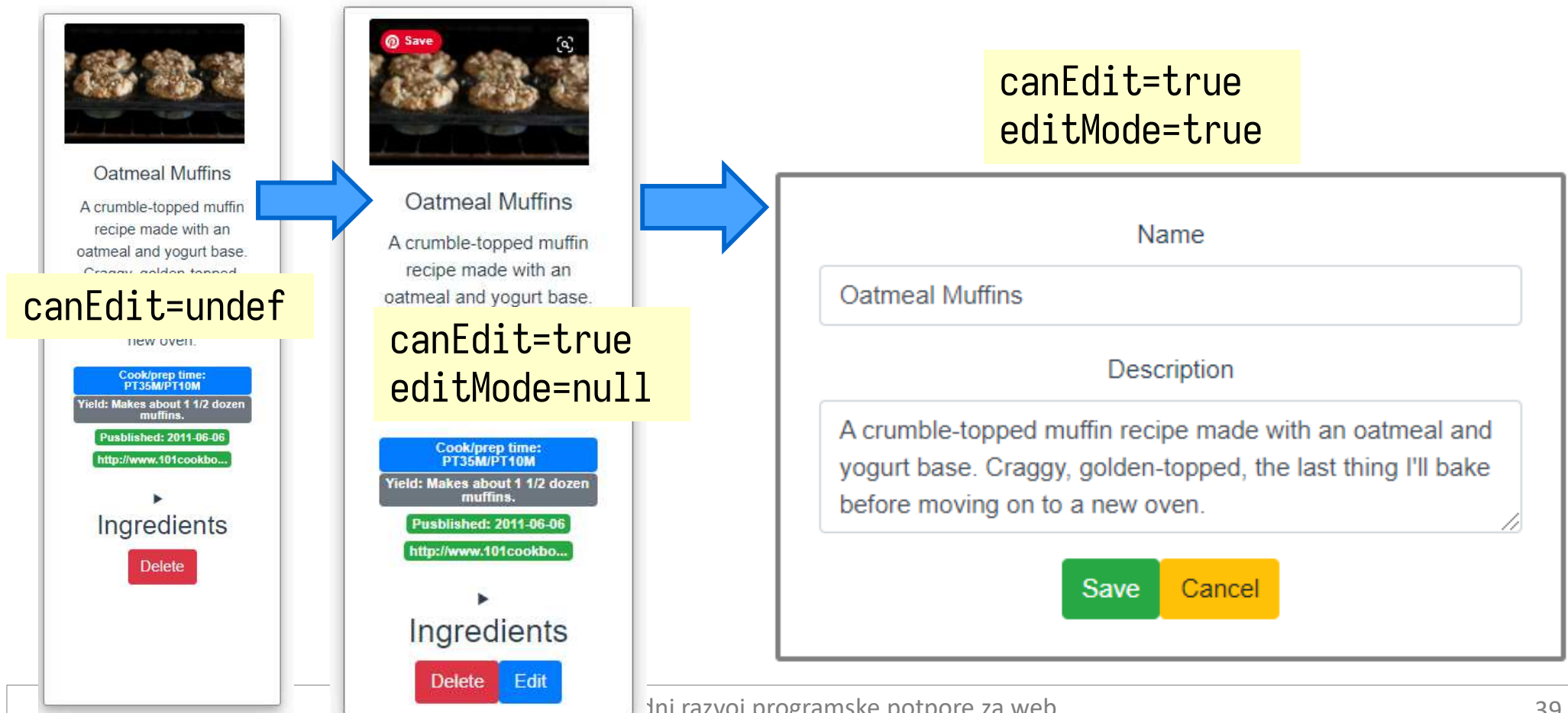
<http://www.101cookbo...>

Ingredients

Delete

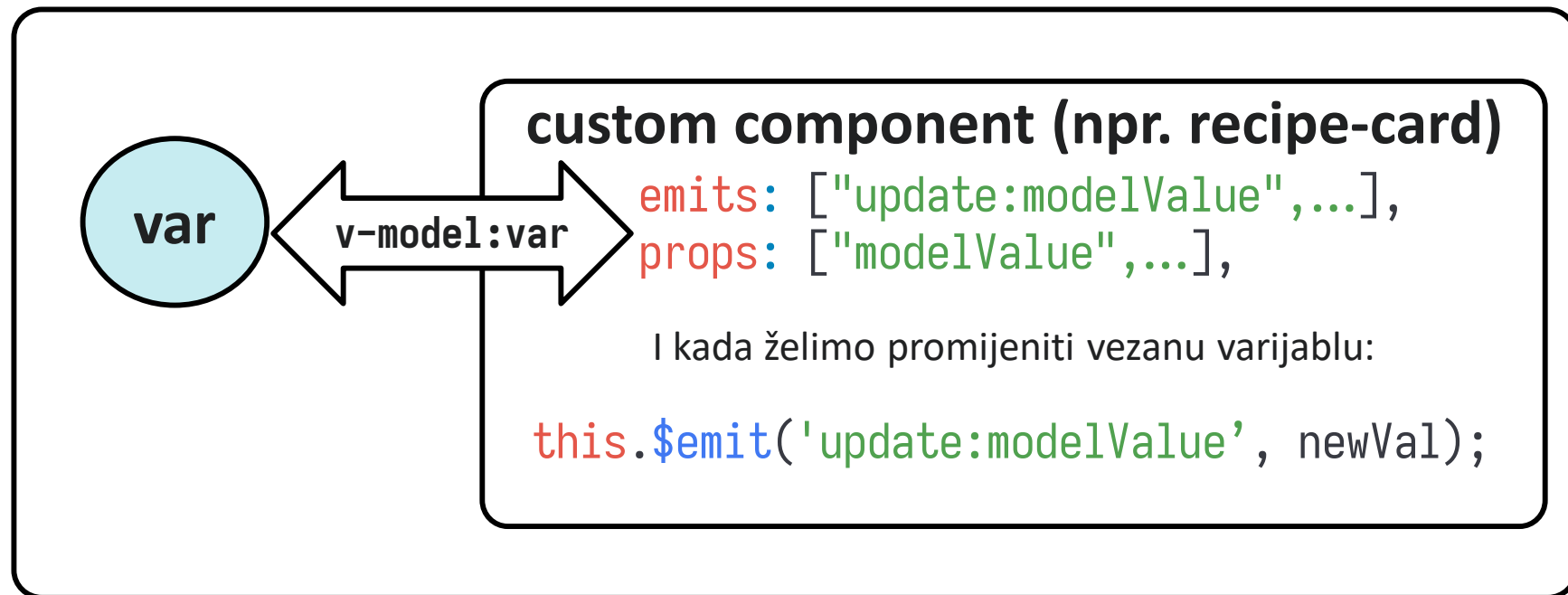
Omogućimo izmjenu recepta!

- Radi jednostavnosti: samo name i description
- Gumb za izmjenu vidljiv samo kada se odabere recept
 - Dodat ćemo property canEdit koji će to definirati
- Komponenta mijenja izgled s obzirom na editMode



Omogućimo izmjenu recepta!

- Želimo da se izmjene unutar komponente odraze na vanjske podatke odnosno selektirani recept
- Gumb za izmjenu vidljiv samo kada se odabere recept
 - Dodat ćemo prop `canEdit` koji će to definirati
- Komponenta mijenja izgled s obzirom na `editMode`



Povežimo prvo „vanjsku” varijablu

views/recipes.vue

```
<template>
  <div v-if="selectedRecipe">
    <div>
      <recipe-card :key="selectedRecipe.id"
        v-model="allRecipes[selectedRecipeIndex]"
        @delete-recipe="deleteRecipe"
        can-edit="true"
      ></recipe-card>
    </div>
  </div>
  <div v-else>
    <div>
      <recipe-card v-for="(recipe, index)
        in allRecipes"
        :key="recipe.id"
        v-model="allRecipes[index]"
        @delete-recipe="deleteRecipe"
      ></recipe-card>
    </div>
  </div>
</template>
...
```

Nema smisla povezati na lokalnu selectedRecipe jer se to onda neće odraziti na naše polje – zato uvodimo selectedRecipeIndex da možemo odmah direktno povezati s elementom polja `allRecipes[selectedRecipeIndex]`

```
...
<script>
export default {
  ...
  data() {
    return {
      selectedRecipe: null,
      selectedRecipeIndex: -1,
    };
  },
  watch: {
    $route(to, from) {
      this.selectedRecipe = this.allRecipes.find( x => x.id == this.$route.params.id);
      this.selectedRecipeIndex = this.allRecipes.findIndex( x => x.id == this.$route.params.id);
    },
  },
  ...
}
```

Ovdje bi nam bili dovoljni props iz ver04 jer se predaju samo roditelj->dijete jer canEdit nije postavljen pa se ne može ni promijeniti vezana vrijednost, ali ovako je bolje jer kasnije možemo slobodno staviti canEdit – probajte!

... zatim recipe-card

views/recipes.vue

```
<template>

  <small-card v-if="!editMode">
    (... isto kao prije ...)
    <button v-if="canEdit" @click="editMode = true">
      Edit
    </button>
  </small-card>
  <div v-if="editMode" class="editForm">
    <form @submit.prevent="submitChanges">
      <input type="text" v-model="name">
      <textarea v-
model="description"></textarea>
      <button type="submit">
        Save
      </button>
      <button @click.stop="exitSingleRecipe()">
        Cancel
      </button>
    </form>
  </div>
</template>
```

Izbačeno dosta koda koji nije u fokusu, vidjeti src, jedino u read-only dijelu sad koristimo modelValue.PROP, npr. modelValue.name, modelValue.url, itd.

Save će bubble-up u form submit, a Cancel neće zbog stop modifera

```
<script>
export default {
  emits: ["deleteRecipe", "update:modelValue"],
  props: ["modelValue", "canEdit", "zoom" ],
  data() {
    return {
      editMode: false,
      name: this.modelValue.name,
      description: this.modelValue.description
    }
  },
  methods: {
    exitSingleRecipe() {
      this.$router.push({ path: '/recipes' });
    },
    submitChanges() {
      this.$emit('update:modelValue', {
        id: this.modelValue.id,
        image: this.modelValue.image,
        name: this.name,
        description: this.description,
        cookTime: this.modelValue.cookTime,
        prepTime: this.modelValue.prepTime,
        recipeYield: this.modelValue.recipeYield,
        datePublished: this.modelValue.datePublished,
        url: this.modelValue.url,
        ingredients: this.modelValue.ingredients
      });
      this.exitSingleRecipe();
    }
  }
  ...
}
```

Slide 42

DM6

kuzim, ali bas mi je update:modelValue grd, moze se spomenuti state model. U biti je li bitno za recipe promjena u kartici?

Danijel Mlinarić; 15.9.2021.

DM7

sad vidim sljedeći slajd spominjes stanja :)

Danijel Mlinarić; 15.9.2021.

Može i bolje...

- Što se dogodi s promjenama ako odemo na Calculator, pa se vratimo na Recipes?

Stanje aplikacije (*state*)

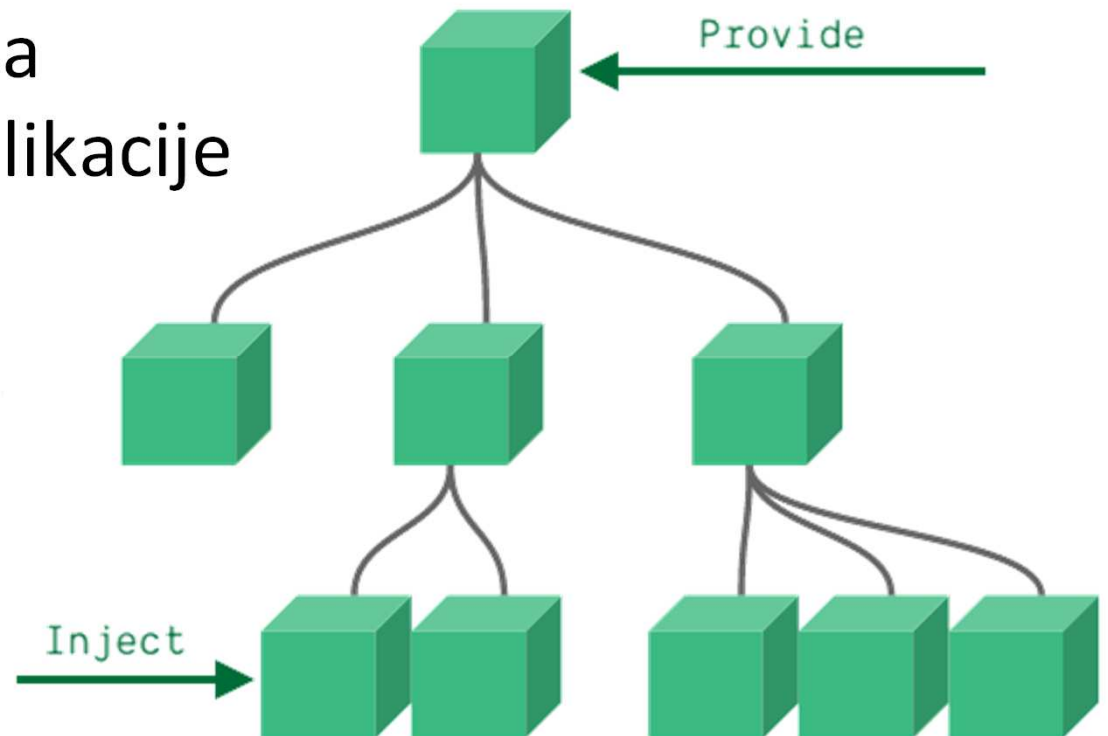


Stanje

- Stanje = trenutni radni skup podataka aplikacije
- Može biti:
 - A. Lokalno stanje** (komponenti, stranica,...)
 - Odnosi se na samo jednu komponentu
 - Npr. logs u Calculator, isEdit u RecipeCard, allRecipes u Recipes
 - B. Globalno stanje**
 - Odnosi se na (potrebno u) više komponenti
 - Npr. sadržaj košarice kod kupovine, prijavljeni korisnik i uloge
 - Je li allRecipes potreban na više mjesta?
- Tipično u aplikaciji imamo i globalno i lokalno stanje

Kako ostvariti globalno stanje? (1/2)

- *Provide/inject?*
- Moguće, ali:
 - U korijensku komponentu bi „nagurali” previše koda
 - Teže upravljivo, teži razvoj
 - Nisu jasni obrasci korištenja, izmjene
 - Podložnije pogreškama
 - Prikladno za manje aplikacije



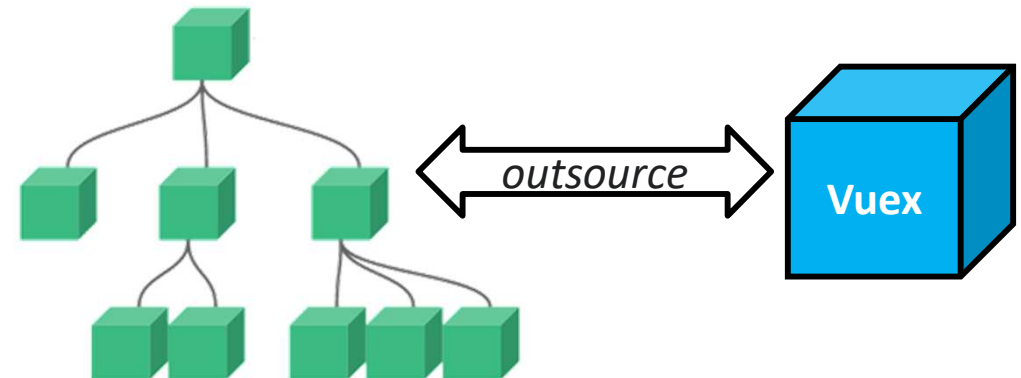
Kako ostvariti globalno stanje?

(2/2)

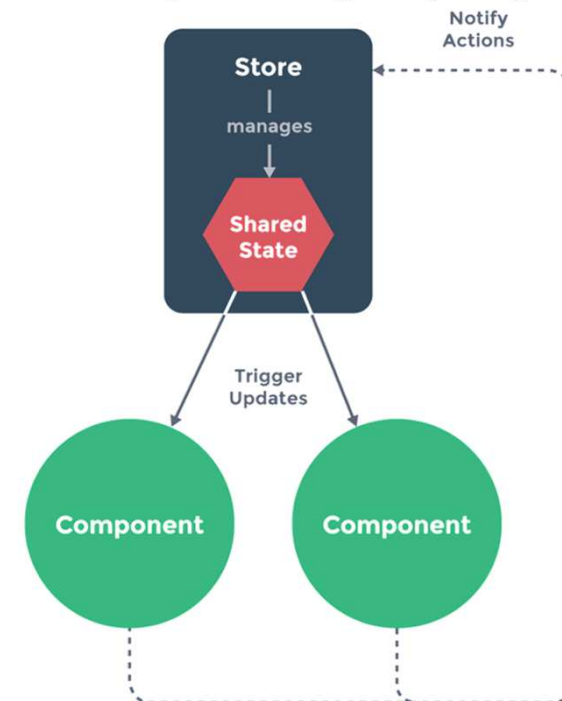
- FTSE 😊: uvodimo vanjski entitet koji se bavi globalnim stanjem

- *Global singleton*

- Provodi nekakav **obrazac** upravljanja stanjem:



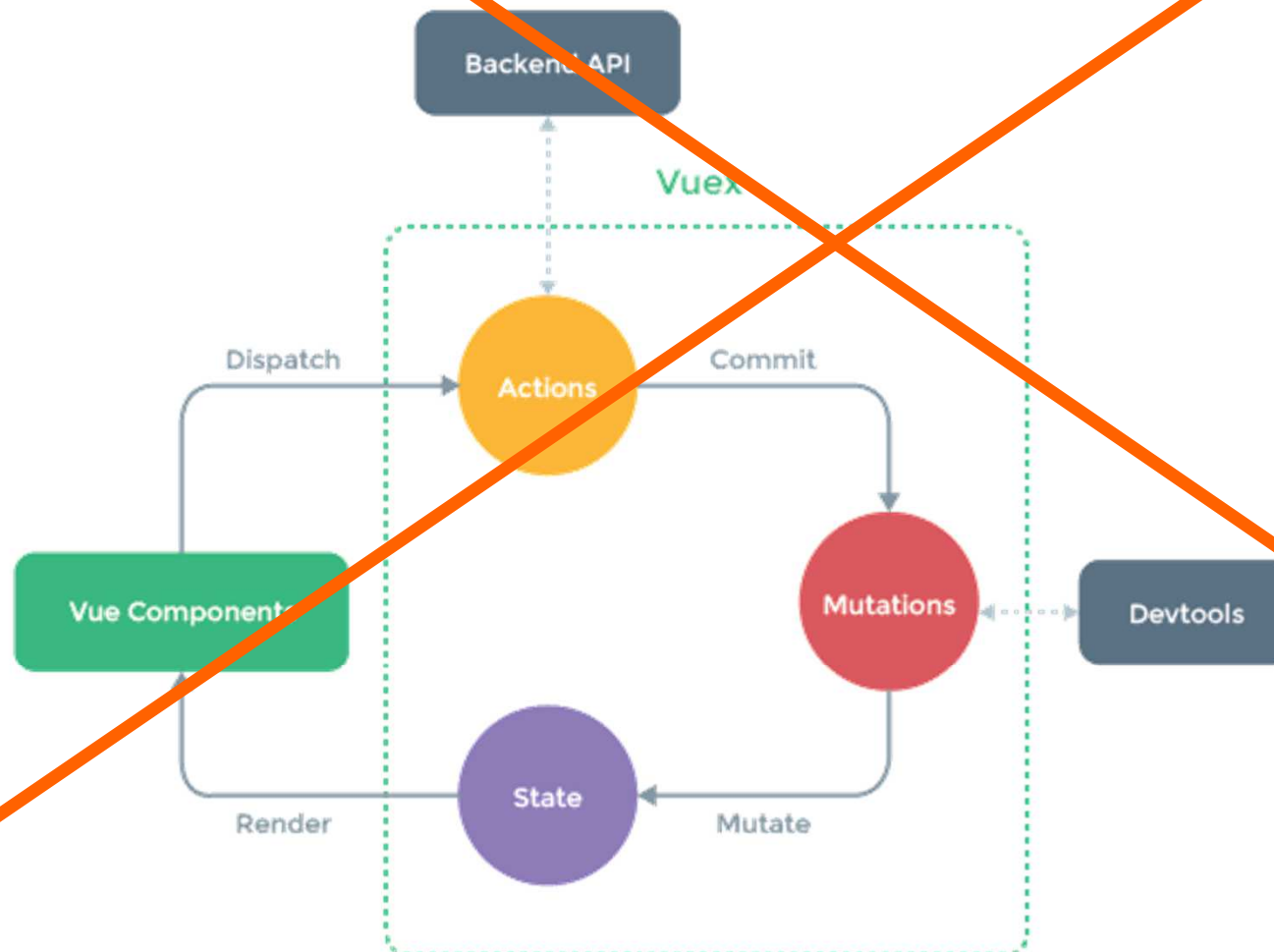
- **Strogo definirani načini korištenja, čitanja i mijenjanja stanja**
- Smanjuje mogućnost pogreške
- Olakšava razvoj (devtools, debugging, time-travel)
- Npr.:
 - Redux, MobX
 - Vuex, Pinia
 - NgRx



<https://v3.vuejs.org/guide/state-management.html#simple-state-management-from-scratch>

■ Vuex

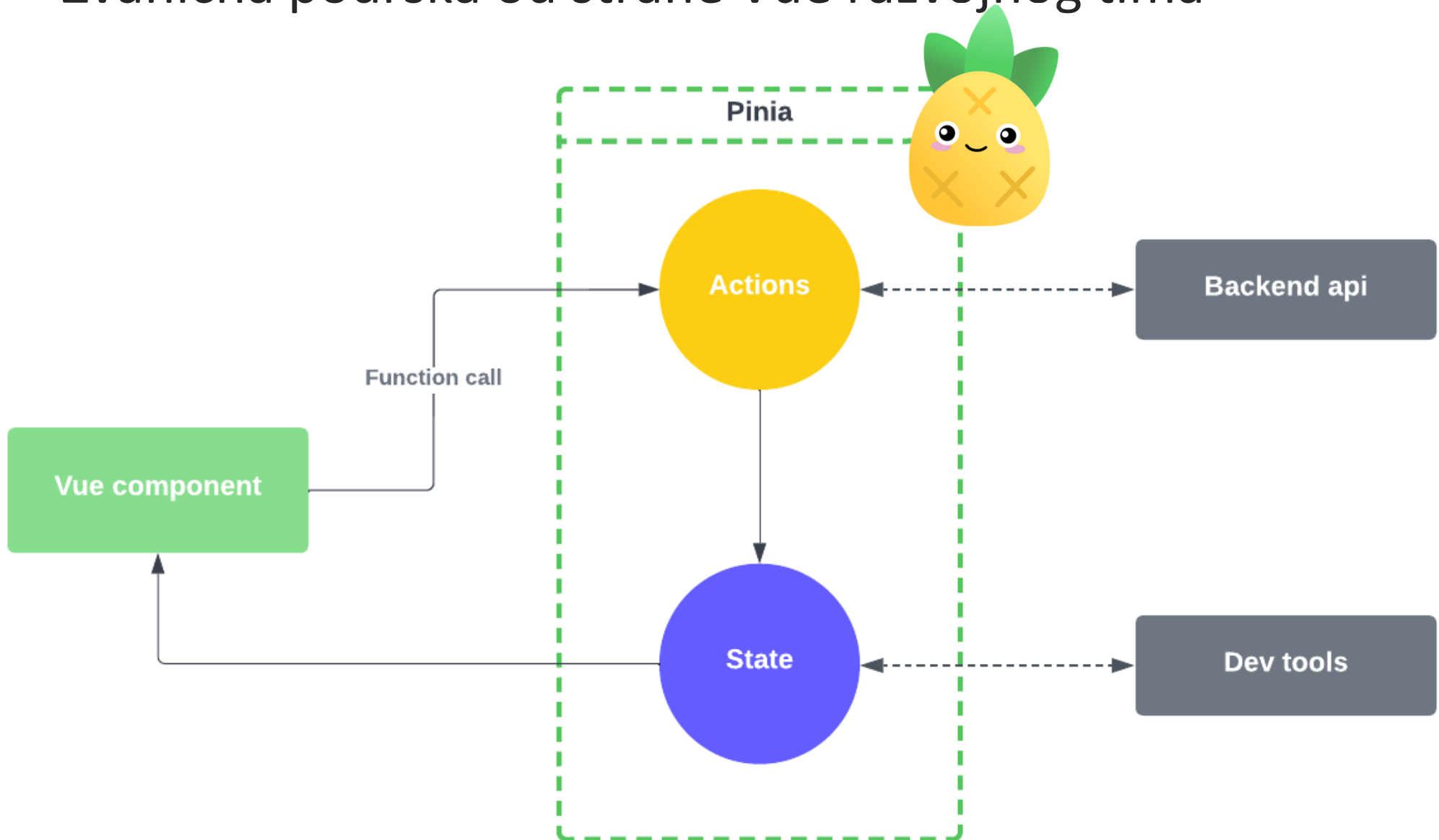
- „state management pattern + library for Vue.js applications”
- „strogo definirani način korištenja” (obrazac):



<https://vuex.vuejs.org/#what-is-a-state-management-pattern>

Pinia

- Zvanična podrška od strane Vue razvojnog tima



<https://betterprogramming.pub/testing-pinia-is-vuex-out-43e0531824f5>

Pinia counter app – rudimentarni primjer

stores/counter.js

```
import { defineStore } from "pinia";
export const useCounterStore = defineStore({
  id: "counter",
  state: () => ({
    counter: 0,
  }),
  getters: {
    count: (state) => state.counter,
    doubleCount: (state) => state.counter * 2,
  },
  actions: {
    increment() {
      this.counter++;
    },
    decrement() {
      this.counter--;
    },
  },
});
```

Count: 1
Double count: 2



state, getters i
actions

App.vue

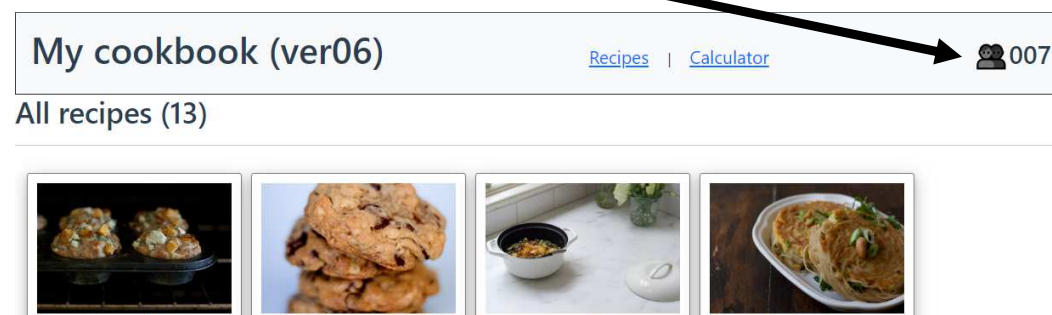
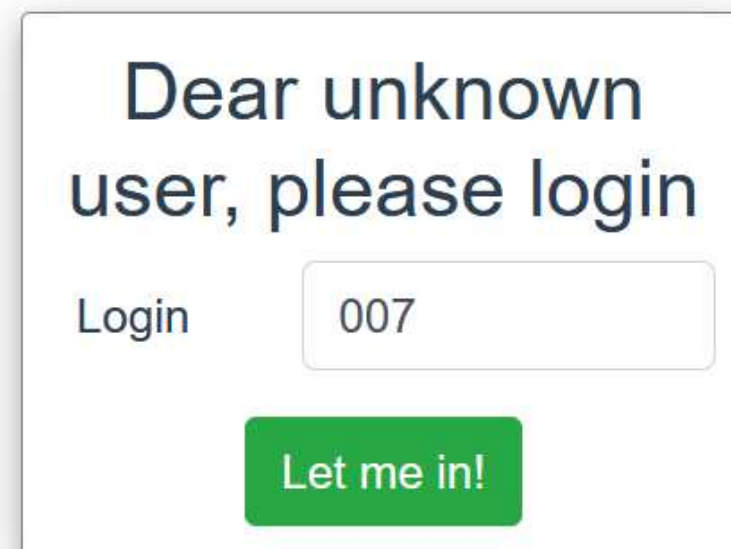
```
<template><div>
  <h1>Count: {{ count }}</h1>
  <h1>Double count: {{ doubleCount }}</h1>
  <p>
    <button @click="increment">+</button>
    <button @click="decrement">-</button>
  </p>
</div></template>
<script>
import { useCounterStore } from "../stores/counter";
import { mapState, mapActions } from "pinia";
export default {
  // data() { return { store: useCounterStore(), }; },
  computed: {
    ...mapState(useCounterStore, ["count",
    "doubleCount"]),
  },
  methods: {
    ...mapActions(useCounterStore, ["increment",
    "decrement"]),
    // decrement() { this.store.decrement(); },
  },
};
</script>
```

Može i komentirana verzija, ali elegantnije je koristiti helpere mapState i mapActions koji će nam zadana svojstva/funkcije mapirati unutra komponente

Dodajmo prijavljenog korisnika!

■ Plan:

- Store sadrži ime prijavljenog korisnika
- Napraviti ćemo novu stranicu **LoginView.vue** za prijavu
- U router/index.js:
 - Ako **nije** prijavljen -> /login
 - Ako je prijavljen – kao prije
- Store ćemo registrirati i on će biti vidljiv svim komponentama, u našem slučaju trebaju ga **router**, **LoginView** i **App** (radi ikonice gore desno)
- **Store pravila:**
 - Čitamo pomoću **getters**
 - Mijenjamo stanje pomoću **actions**



Store

stores/index.js

```
import { defineStore } from "pinia";

export const useAuthStore = defineStore("auth", {
  state: () => ({
    // ovo bi zapravo bio neki objekt, ali ovdje
    // radi jednostavnosti je user=username
    user: null,
    _landingUrl: "/",
  }),
  getters: {
    isAuthenticated: (state) => !!state.user,
    username: (state) => state.user,
    landingUrl: (state) => state._landingUrl,
  },
  actions: {
    setUsername(username) {
      this.user = username;
    },
    setLandingUrl(url) {
      this._landingUrl = url;
    },
  },
});
```

main.js

```
...
import { createApp } from "vue";
import { createPinia } from "pinia";
...
const app = createApp(App);

app.use(createPinia());
app.use(router);
```

Stvaramo singleton i vežemo ga uz aplikaciju. Kasnije, u komponentama, ćemo ga jednostavno referencirati s:

```
import { useAuthStore } from "../stores/auth";
...
const auth = useAuthStore();
```

Router

Router/index.js

```
import { useAuthStore } from "../stores/auth";
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/login",
      name: "login",
      component: LoginView,
    },
    {
      path: "/", name: "recipes", component: RecipesView,
    },
    ...
  ],
  router.beforeEach(async (to) => {
    // redirect to login page if not logged in and trying to access
    // a restricted page
    const publicPages = ["/login"];
    const authRequired = !publicPages.includes(to.path);
    const auth = useAuthStore();
    if (authRequired && !auth.isAuthenticated) {
      auth.setLandingUrl(to.fullPath);
      return "/login";
    }
  });
});
```

Dodajemo novu rutu na stranicu za prijavu korisnika (sljedeći slajd)

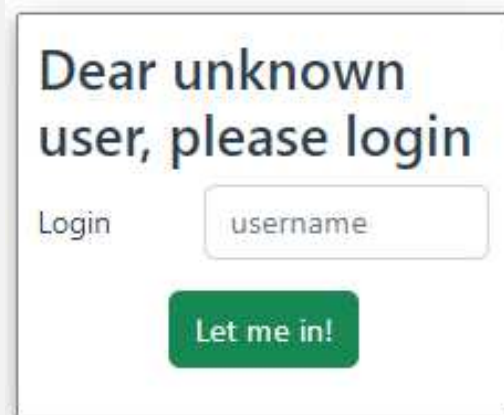
„Lovimo” sve adrese i prije usmjeravanja provjeravamo je li potrebna prijava

Pamtimo traženi URL da bi mogli korisnika preusmjeriti nakon uspješne prijave

UserLogin.vue

views/LoginView.vue

```
<template>
  <small-card>
    <h3>Dear unknown user, please login</h3>
    <form @submit.prevent="login">
      <div class="form-group row">
        <label class="col-4 col-form-label">Login</label>
        <div class="col-8">
          <input class="form-control"
            placeholder="username"
            v-model.trim="username"
          />
        </div>
      </div>
      <div class="d-flex justify-content-center">
        <button class="btn btn-success m-3"
          type="submit">
          Let me in!
        </button>
      </div>
    </form>
  </small-card>
</template>
```

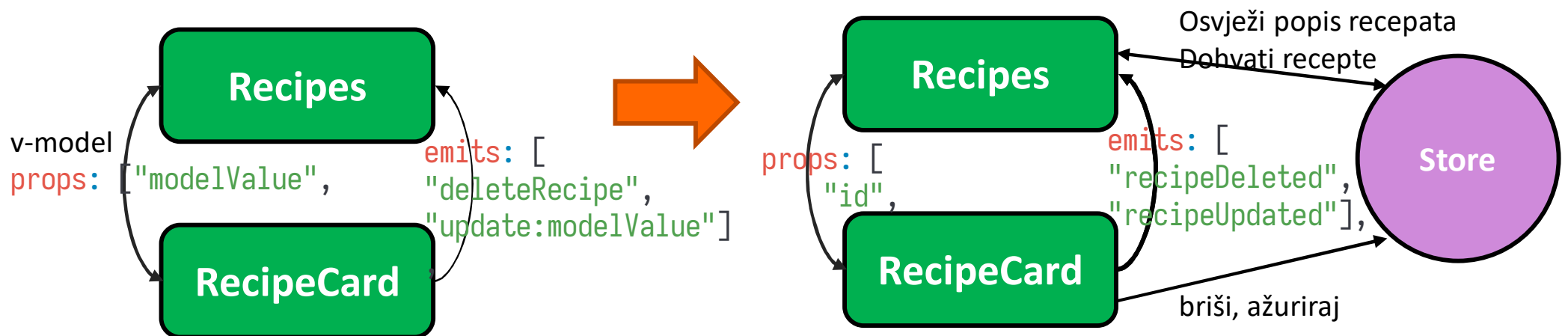


```
<script>
import { mapActions, mapState } from "pinia";
import { useAuthStore } from "../stores/auth";
export default {
  data() {
    return {
      username: "",
    };
  },
  computed: {
    ...mapState(useAuthStore, ["landingUrl"]),
  },
  methods: {
    ...mapActions(useAuthStore, ["setUsername"]),
    login() {
      // sve puštamo, naravno inače treba provjeriti
      this.setUsername(this.username);
      this.$router.push({ path: this.landingUrl });
    },
  },
};
</script>
```

Vraćamo korisnika na
inicijalno zatraženi
URL

Izmjene ostatka aplikacija – drugi store *recipes*

- Pinia jednako tretira/dopušta sinkrone i asinkrone funkcije pa ćemo tako u recipe store premjestiti
 - Dohvat recepata (**asinkrona** funkcija)
 - Brisanje i ažuriranje recepata (sinkrone funkcije)
- Promijenit ćemo ustroj i povezivanje **Recipes** i **RecipeCard**:
 - prije** su bili spregnuti putem v-model, emit, refresh iz **Recipes**
 - sada** obje imaju pristup storeu i **Recipe** predaje samo **id**



Pogledajmo prvo *Recipe store*:

store/recipes.js

```
export const useRecipeStore = defineStore("recipes", {
  state: () => ({
    _allRecipes: [],
  }),
  getters: {
    allRecipes: (state) => state._allRecipes || [],
    getRecipeById: (state) => {
      return (id) => state._allRecipes.find((rcp) => rcp.id === id);
    },
  },
  actions: {
    async refreshRecipes() {
      if (this.allRecipes.length === 0) {
        try {
          ...
          this._allRecipes = recipes.slice(0, 50);
        } else { ... }
      } catch (error) { ... }
    },
    deleteRecipe(id) {
      this._allRecipes = this._allRecipes.filter((x) => x.id !== id);
    },
    updateRecipe(recipe) {
      this._allRecipes = this._allRecipes.map((x) =>
        x.id === recipe.id ? recipe : x
      );
    },
  },
});
```

Trebat će nam za
RecipesView

Trebat će nam za
RecipeCard, primjer
gettera s argumentima

Getters are just computed properties behind the scenes, so it 's not possible to pass any parameters to them. However, you can return a function from the getter to accept any arguments:

Trivijalno keširanje, razmisliti o
opcijama sinkronizacije s *backendom*...
Svakako dodati boolean da se može
forsirati refresh...

... zatim RecipeCard - on radi samo na temelju

id i canEdit (i veze sa Storeom), mapActions, mapState:

ver06

components/RecipeCard.js

```
export default {
  emits: ["recipeDeleted", "recipeUpdated"],
  props: ["id", "canEdit", "zoom"],
  data() { return {
    editMode: false, recipe: { url: "" },
  }; },
  computed: {
    ...mapState(useRecipeStore, ["getRecipeById"]),
    idUrl() { return "/recipes" + this.id; },
  },
  methods: {
    ...mapActions(useRecipeStore, ["deleteRecipe", "updateRecipe"]),
    deleteRecipeLocal() {
      this.deleteRecipe(this.id); this.$emit("recipeDeleted", { id: this.id });
      this.exitSingleRecipe();
    },
    exitSingleRecipe() { this.$router.push({ path: "/recipes" }); },
    submitChanges() {
      this.updateRecipe(this.recipe); this.$emit("recipeUpdated", this.recipe);
      this.exitSingleRecipe();
    },
  },
  async created() {
    this.recipe = { ...this.getRecipeById(this.id) };
  },
};
```

Template je isti, samo sada koristimo recipe.property, npr.

```
<span class="badge badge-secondary">Yield: {{ recipe.recipeYield }}</span>
```

Također, u edit formi smo izbacili vlastite varijable već vežemo direktno na recipe.name i recipe.description – to je u redu jer je taj recipe klonirani element polja iz allRecipes, dakle lokalna kopija, npr.

```
... <div v-if="editMode" class="editForm">
  <form @submit.prevent="submitChanges">
    ...
    <input type="text" class="form-control" id="name" placeholder="name" v-model="recipe.name">
```

Ljubaznost: možda će postfestum zanimati onoga tko me pozvao. Primijetiti promjenu naziva eventa - sad je pasiv

Inicijalno prazan, korisnik to neće ni vidjeti, ali na jedno mjestu koristimo url.substring, pa da ne bi bilo undefined.substring

Kloniramo putem spread operatora.
Mogli smo i u mounted, ali created je ranije u lifecycleu

... i konačno - RecipesView

views/RecipesView

```
export default {
  props: ["id"],
  data() {
    return { selectedRecipe: null };
  },
  computed: {
    ...mapState(useRecipeStore, ["allRecipes"]),
  },
  methods: {
    ...mapActions(useRecipeStore, ["refreshRecipes"]),
    recipeUpdated(recipe) {
      console.log("So now i know recipe is updated...", recipe);
    },
    recipeDeleted(args) {
      if (this.selectedRecipe
        && this.selectedRecipe.id === args.id) {
        this.selectedRecipe = null;
      }
    },
  },
  async mounted() {
    await this.refreshRecipes();
    this.setSelectedRecipe();
  },
};
```

Recipes se bitno pojednostavnio
– sad se brine samo time je li
recept odabran ili ne i kako to
prikazati

Ažuriranje nas zasad ni ne zanima, ali
brisanje da – koristimo „ljubaznost“
odnosno dojavu iz RecipeCard

```
<div v-if="selectedRecipe">
  <h2>Recipe #{{ id }}</h2><br>
  <div class="d-flex justify-content-center">
    <recipe-card :key="selectedRecipe.id"
      :id="selectedRecipe.id"
      @recipe-updated="recipeUpdated"
      @recipe-deleted="recipeDeleted"
      can-edit="true"
    ></recipe-card>
  </div>
</div>
<div v-else>
  <h2>All recipes ({{$store.getters.allRecipes.length}})</h2>
  <hr>
  <div class="container-fluid p-2 d-flex flex-wrap">
    <recipe-card v-for="recipe in allRecipes"
      :key="recipe.id"
      :id="recipe.id"
      @recipe-updated="recipeUpdated"
      @recipe-deleted="recipeDeleted"
    ></recipe-card>
  </div>
</div>
```

DZ: promijeniti template da
pokazuje spinner dok recepti
još nisu dohvaćeni...

I još jedna stvar...

npm run build



```
\ Building for production...
```

```
WARNING Compiled with 1 warning  
warning
```

```
4:46:18 PM
```

```
asset size limit: The following asset(s) exceed the recommended size limit (244 KiB).  
This can impact web performance.
```

```
Assets:
```

```
img/disco.b921cfd5.gif (2.06 MiB)
```

File	Size	Gzipped
dist\js\chunk-vendors.a68e7e1d.js	142.14 KiB	50.69 KiB
dist\js\app.a448ee5c.js	15.79 KiB	5.00 KiB
dist\js\chunk-2d0ddf93.8177cced.js	0.48 KiB	0.33 KiB
dist\js\chunk-91eb2748.2e907fc5.js	0.42 KiB	0.30 KiB
dist\css\app.1961192e.css	0.52 KiB	0.34 KiB

```
Images and other types of assets omitted.
```

```
DONE Build complete. The dist directory is ready to be deployed.
```

```
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

Razvojna i produkcijska okolina

- U razvojnoj okolini:
 - Pokreće se lokalni web-poslužitelj
 - Prevodi se .vue kontinuirano kod svakog snimanja
 - Hot-Module-Replacement (HMR)
 - Datoteke nisu minimizirane
 - Radni okvir daje raskošne provjere i upozorenja
 - Itd.
- U produkcijskoj okolini:
 - SPA je zapravo „samo” HTML + CSS + JS koji zatim trebamo poslužiti pregledniku (static web hosting)
 - Minifikacija HTML+CSS+JS datoteka (+ source maps)
 - Vendor chunk splitting (razdvojiti naš kod od knjižnica)
 - Optimizacija koda
 - Optimizacija učitavanja?

Optimizirajmo učitavanje (dijelova) stranice

- Jedan od nedostataka SPA je veliki inicijalni zahtjev
- **Rješenje: učitajmo dijelove aplikacije kad (i ako trebaju)!**
- Dodajmo u našu aplikaciju besmislenu „Disco” komponentu i stranicu:



- Učitajmo ju asinkrono – kad (ako) nam zatreba!

Komponenta i stranica su trivijalne:

views/DiscoView.vue

```
<template>
  <div>
    <h2>Disco!!!</h2>
    <div class="container-fluid p-2 d-flex justify-content-center">
      <disco-component></disco-component>
    </div>
  </div>
</template>
```

components/DiscoComponent.vue

```
<template>
  <div>
    
  </div>
</template>
```


Prvo učitajmo komponentu asinkrono...

ver07

main.js

```
import { createApp, defineAsyncComponent } from "vue";
import App from "./App.vue";
import store from "./store";
import router from "./router";
import RecipeCard from './components/RecipeCard.vue';
import TempConverter from './components/TempConverter.vue';
import SmallCard from './components/SmallCard.vue';

const DiscoComponent = defineAsyncComponent(() => import('./components/DiscoComponent.vue'));

const app = createApp(App);
app.use(store);
app.use(router);
app.component('recipe-card', RecipeCard);
app.component('temp-converter', TempConverter);
app.component('small-card', SmallCard);
app.component('disco-component', DiscoComponent);
app.mount("#app");
```

...potom i stranicu

router/index.js

```
import { defineAsyncComponent } from "vue";
import { createRouter, createWebHistory } from "vue-router";
import Recipes from "../views/Recipes.vue";
import Calculator from "../views/Calculator.vue";
import NotFound from "../views/NotFound.vue";
// u routeru ne koristiti defineAsyncComponent
const Disco = () => import('../views/DiscoView.vue');
const routes = [
  {
    path: "/",
    component: Recipes, // mogli smo i redirect: "/recipes"
  }, ...
  {
    path: "/disco",
    component: Disco,
  },
  ...
];
```

Ako pokrenemo i u razvojnoj okolini vidjet ćemo da se datoteke dohvaćaju kada su potrebne.
Kada napravimo production build, onda će to biti posebne chunk datoteke.

Konačno, napravimo build

■ npm run build

```
λ npm run build
```

```
> fer-demo-spa@0.0.0 build
> vite build
```

```
vite v3.2.5 building for production...
```

```
✓ 45 modules transformed.
```

```
dist/assets/disco.0d123211.gif
```

```
2112.26
```

```
dist/assets/travolta404.0518648b.gif
```

```
2678.9 KiB
```

```
dist/index.html
```

```
0.70 KiB
```

```
dist/assets/DiscoComponent.4c4e8f93.js
```

```
0.22 KiB / gzip: 0.19 KiB
```

```
dist/assets/DiscoView.c1014e9c.js
```

```
0.30 KiB / gzip: 0.23 KiB
```

```
dist/assets/index.40af8dc2.css
```

```
1.99 KiB / gzip: 0.79 KiB
```

```
dist/assets/index.95112b1f.js
```

```
89.54 KiB / gzip: 35.13 KiB
```

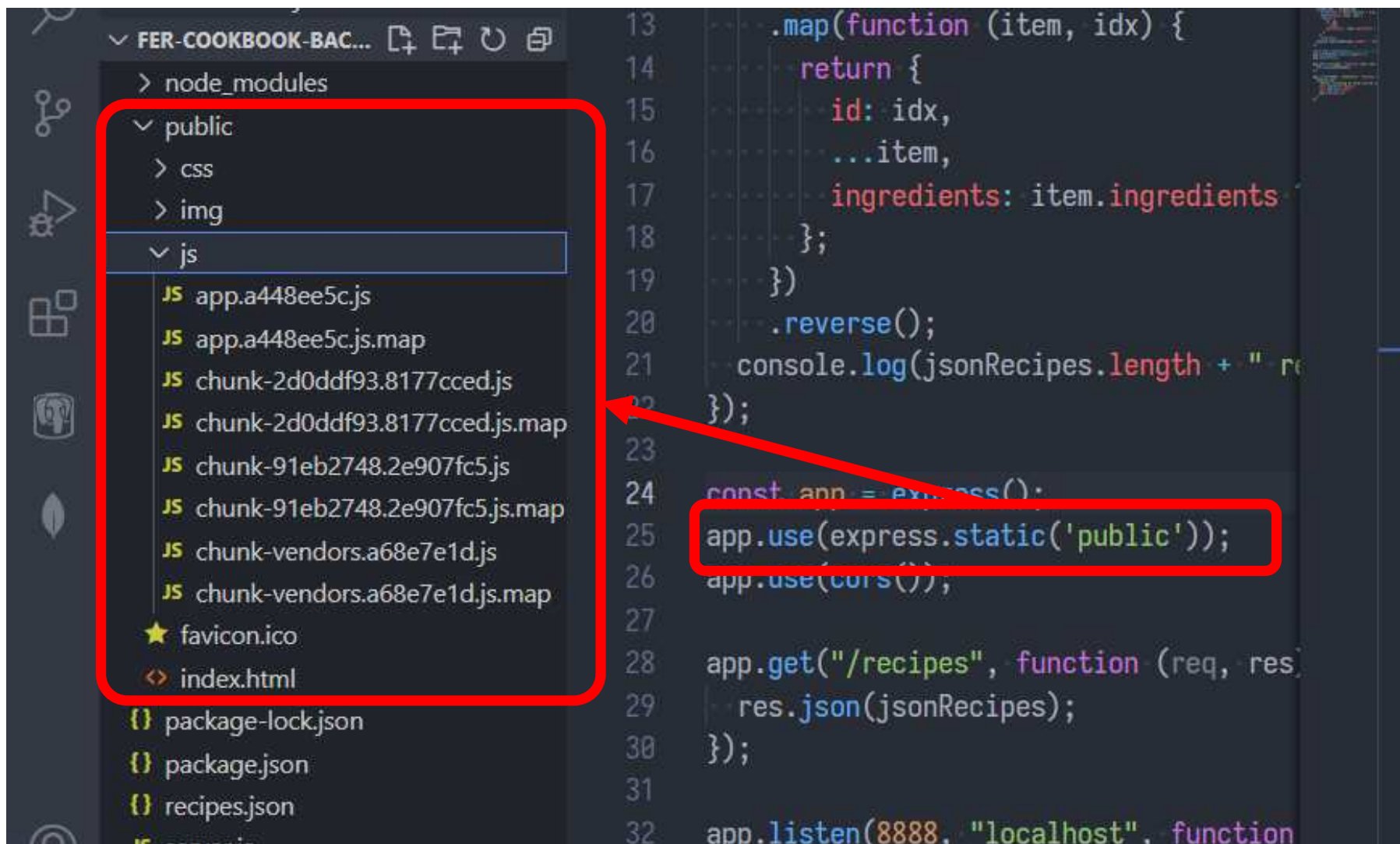
U razvojnoj okolini:

Name	Method	Status	Type	Initiator	Size	T
app.js	GET	200	script	(index)	274 kB	1
chunk-vendors.js	GET	200	script	(index)	3.0 MB	6

Ovisnosti (tuđi kod) u posebnoj datoteci, naš kod u main.js, preostali dijelovi (asinkroni) našeg koda u chunk datoteka i minificirani CSS.

Poslužimo putem naše backend aplikacije

- Kopiramo /dist direktorij koji je napravio vue-cli u /public folder backend



■ Projekt

- Napraviti SPA aplikaciju u Vue3 radnom okviru
- *Peer assessment* putem Edgara
- Više u Edgaru...

Komentar: Composition API

- U primjerima je korišten jednostavniji „Options API”
- Počev od v2.7. moguće je koristiti alternativni „Composition API”
- Primjer s <https://vuejs.org/guide/extras/composition-api-faq.html#what-is-composition-api>

The setup attribute is a hint that makes Vue perform compile-time transforms that allow us to use Composition API with less boilerplate. For example, imports and top-level variables / functions declared in <script setup> are directly usable in the template.

```
<script setup>
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)

// functions that mutate state and trigger updates
function increment() {
  count.value++
}

// lifecycle hooks
onMounted(() => {
  console.log(`The initial count is ${count.value}.`)
})
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

■ Zašto *Composition API*?

- Better Logic Reuse
 - Primarna prednost: ponovna uporaba programske logike – odatle composition
 - Pogledati npr. <https://vueuse.org/>
- More Flexible Code Organization
 - Slika desno ->
- Bolji type inference, ts.
- Efikasniji i manji konačni kod
- **Nota bene:**
 - Mogu se kombinirati Options + Comoposition
 - Options neće biti deprecated!

Options API



```
export default {
  data() {
    return {
      count: 0,
    }
  },
  methods: {
    increment() {
      this.count++
    },
  },
  computed: {
    doubleCount() {
      return this.count * 2
    },
  },
}
```

Composition API



```
import { ref } from 'vue'

export default {
  setup() {
    const count = ref(0)

    function increment() {
      count.value++
    }

    const doubleCount = computed(() => count.value * 2)

    return {
      count,
      increment,
      doubleCount,
    }
  },
}
```

Composition API

- Nećemo ga detaljno objašnjavati dalje
- U projektu možete koristiti bilo koji, a mogu se i kombinirati
- Za one koji se žele znati više o Composition API-ju, pogledati:
 - Prerađenu verziju konačnog primjera ver07-composition-api
 - Vue dokumentaciju, podesite *API preference*:

